jrg:
nr:

AUGUSTUS 1983

++

- ACORN Nieuws -

++

3000 BAUD!

S CHAKELKAART

O PERATING

S YSTEMEN



GRAPHICS

++

ACORN NIEUWS Uitg.Fed.Acorn Computerclubs Ned/B.

Verschijnt 6-8 x p/jr.

UIT DE FEDERATIE.

Even een aantal zaken rechtzetten c.q. uit de wereld helpen.
Er schijnen een aantal misverstanden te zijn omtrent het gebruik van ledenlijsten.
Ten eerste is de federatie een overkoepelend orgaan die niet meer dan vijftien verenigingen als lid heeft. Dit impliceert dat de federatie niet vrij is om ledenlijsten van aangesloten verenigingen te verspreiden. Alleen de penningmeester heeft een ledenlijst die compleet is en deze is nodig voor het bijhouden van de contributie inning en verspreiding van het Acorn nieuws. Buiten een aantal juridische obstakels als wel om praktische redenen is het niet mogelijk om de ledenlijst te verspreiden. Dit zou inhouden dat er iedere week een ledenlijst zou moeten worden gemaakt, daar er ongeveer tien leden per week bijkomen. Indien iemand van mening is dat hij de ledenlijst nodig heeft, dan staat het hem of haar geheel vrij de veertien andere verenigingen te benaderen om een ledenlijst. Op dat moment is de verantwoordelijkheid bij de desbetreffende vereniging, zolang die vereniging de ledenlijst maar niet commerciëel exploiteerd.

Een ander punt is mijn inziens belangrijker daar dit de aard van de vereniging zelf aantast. Het gaat hier om het commerciëel ter beschikking stellen van zelf ontwikkelde soft/hardware. Ik vind het ronduit schandalig als iemand eens iets presteert en er dan maar direct een slaatje uit probeert te slaan, wat de argumenten daar dan ook voor mogen zijn. Deze mensen staan er kennelijk niet bij stil dat zij wel van alle kennis van anderen op de hoogte zijn zonder noemenswaardige kosten terwijl er hier mensen bij zijn die wezenloos druk zijn voor de club maar er geen cent beter van worden. Deze positieve groep mensen redeneren namelijk als volgt : ten eerste hebben zij er plezier in om iets produktiefs te ontwikkelen en dit voor algemeen belang ter beschikking te stellen, immers zij krijgen d.m.v. het Acorn nieuws, schakelkaart, geheugenkaart etc. er ook weer wat voor terug, het is een kwestie van geven en nemen ! Het begint namelijk schering en inslag te worden dat mensen er zonodig beter van moeten worden.

Ik verwacht geen vergoeding voor mijn diensten maar ik verwacht ook niet te moeten betalen (uitgezonderd algemene kosten) voor iemand anders zijn produktiviteit. Indien dit rottingsproces zich voortzet, zie ik de toekomst erg somber in. We zijn ten-slotte een hobby-club en geen IBM-gebruikersgroep. Ik heb in mijn betoog geen namen genoemd, maar indien men zich aangesproken voelt, lijkt het mij een goede zaak er eens over na te denken hoe verder te gaan. We maken allemaal wel eens fouten maar het is nooit te laat ze toe te geven.

Het volgende punt betreft de toekomst van de ACORN ATOM. Er zijn inmiddels een aantal kaarten op euro formaat gemaakt en is een backplane gemaakt naar Elektuur standaard. Dit backplane heeft 16 slots en is uitbreidbaar tot een veelvoud hiervan. Indien er meer leden zijn die met deze ontwikkeling bezig zijn, zou ik graag een schriftelijke reactie tegemoet zien. Hierop kom ik in het volgende Acorn nieuws uitgebreid terug.

A. Millenaar, Voorzitter

REDACTIONEEL

Hoewel dit nummer in de vakantieperiode is ontstaan, vertoont het geenzins de kenmerken van deze (komkommer-)tijd: 80 pagina's, een record! En nòg is de koek niet op, zie de rubriek "Gezien in andere Acorn-bladen".
Gedeeltelijk mijn eigen schuld, want enkele artikelen zijn min of meer op mijn uitnodiging ontstaan: dank dus aan: Erik Dukker, Bram Poot, Klaas de Raad en Frans van Hoesel. Ook de "gewone" auteurs uiteraard.

Verder verdient melding de verschijning van het eerste nummer van: ACORNTJESBROOD, lijfblad van de regio Twente. 't Zag er bijzonder gedegen uit. Alle succes toegewenst aan redacteur Will Verhoeven! (A propos, Will, denk eens aan de landelijke bladspiegel van 170 x 250mm, wil je ?)

Het blad van de regio Noord heeft een redactie-wisseling ondergaan (sterkte, Jaap Woldringh!) en een naamswisseling, nl. STACKER (spreek uit: "stekker") inplaats van "Acorn Nieuws Noord". Deze naam houdt verband met een elektrisch verbindingsmiddel (stekker), bewaarplaats van informatie (stack) en misschien met de redacteur in zijn moeilijke redactionele momenten.

Verder heeft de regio Noord, in casu Rudi van Drunen, evenals Jaap van de Woestijne enige tijd geleden, het principeschema uitgeplozen van de 16K-kaart.
Uitsluitend voor eerbare doeleinden te bekomen bij Jaap Woldringh.

Tenslotte: nog niet alle regio's hebben gereageerd op mijn "nagekomen bericht" in A.N. 3, betreffende een eventuele uitgave van de 1e jaargang van Acorn Nieuws. Vermoedelijk, zoals regio Noord schreef, omdat de eerstvolgende regio-bijeenkomst pas na de lopende vakantie-periode is gepland. Dom van mij, natuurlijk. Aanmeldings -periode verlengd tot medio september!

Het eerstvolgende nummer van Acorn Nieuws zal o.a. gaan over Analoog/Digitaal omzettingen.
Copy, graag vóór medio september, naar ondergetekende.

Ton Otten (redacteur), J.A. de Gravenlaan 17, 2381 TA Zoeterwoude.

++

BRAND-punten.

Onder deze naam (die ikzelf maar bedacht heb; wie een betere weet, mag het zeggen) wordt U voortaan op de hoogte gehouden van die Atom-aspecten, die in één of meer regio's speciale belangstelling genieten. Het blijkt nl. dat de artikelen, die voor plaatsing in Acorn Nieuws worden ingezonden, meestal gaan over voltooide projecten. Gelukkig niet altijd, maar vaak toch wel. Eigenlijk wel logisch, want schrijven over vage plannen is nu eenmaal moeilijker dan schrijven over iets dat werkt en waar je trots op bent.
Nadeel van deze gewoonte is evenwel, dat de Acorn-gemeenschap pas kennis kan nemen van een bepaald project als het (allang) klaar is terwijl men zelf misschien al een hele tijd aan een soortgelijke ontwikkeling bezig is. Terwijl men SAMEN vermoedelijk sneller had kunnen werken èn wellicht een beter resultaat bereiken! U begrijpt'tal? Inderdaad: SAMEN (nog) WIJZER.

In deze rubriek zullen ontwikkelingen, die in een bepaalde regio de belangstelling hebben, worden opgenomen zodat eenieder die denkt hierin een bijdrage te kunnen leveren, kan meespelen. Kontakten kunnen altijd gelegd worden via de regionale coördinatoren.

Daar gaan we dan, in willekeurige volgorde:

- Regio Noord
  Schakelkaart Operating Systeem, dit A.N., pag.49
  Mini-schakelsysteem, zie "Ontwikkelingen", pag.60
  Real-time klok, zie "Big Benny", pag.45
  Disk Operating System (TELEC)
  RTTY- en Morse-interface
  Eenvoudige numerieke toepassingen, zoals benaderen van funkties door veeltermen en eenvoudige rekenkundige leerprogramma's

- Regio Den Haag.
  Disk Operating Systems
  Schakelkaart Operating System (van Klaas de Raad, beschreven op pag.49)
  2e Video-pagina
  Real Timer (National MM58167A)

- Regio Twente
  80-koloms VDU-kaart
  NMI- en IRQ-schakelingen
  40-koloms VDU-programma
  Schakelsoft
  Database-programma's
  Lineaire analyse van Elektronische Schakelingen
  6502-symbolische assembler
  Twentse Toolkit
- Regio Brabant-Oost
  Viditel*
  Modem
- Regio België
  Behalve 16K-kaart, Schakelkaart en EPROM-programmer:
  Mini Data Cassette Recorder (MDCR-Philips)
  Kleuren-monitor
  A/D- en D/A-converters en toepassingen.
- Regio Overijssel-Gelderland.
  (Zal pas informatie kunnen verstrekken na eerste regionale bijeenkomst na de vakanties.)

Bij dit laatste sluiten zich de overige regio's kennelijk stilzwijgend aan. (Hoop ik, red.)

Tenslotte, à titre personel roept Rob van Dort gegadigden op voor BASICODE-2. Wie wil de Atom-aanpassing aan deze universele computertaal van ons aller NOS (helpen) ontwikkelen? Leest u dan eerst het artikeltje van Rudi van Drunen over dit onderwerp, op pag. 68. Bestelt u dan eventueel boek en bandje (zonder Atom aanpassing!) bij het Algemeen Secretariaat v.d. NOS, te Hilversum, postgiro nummer 1419. Kosten ƒ 25,--.
Bent u dan nog steeds geinteresseerd, neem dan contact op met de voorzitter van het Software Overleg, Roel Heuvel, Brialmontstr. 16 5913 HJ Venlo, tel. 077-40160.

* Hier komt een artikel over, van uw eigen specialist op dit terrein, nl. mijzelf, red.

<<ATOM DESIGN>>
BRON: YOUR-COMPUTER
OMGEZET NAAR ATOM BASIC: W.VERHOEVEN

Dit programmatje genereert allerlei leuke grafische patronen.
Probeer maar uit !

```
10 CLEAR4
20 Y=A.R.%70+100;X=A.R.%60;Z=A.R.%3-4
30 F.R=Y TO X S.Z
40 MOVE R,R;PLOT2,0,(175-2*R)
50 PLOT2,(255-2*R),0;PLOT2,0,(-175+2*R);PLOT2,(-255+2*R),0
60 NEXT
70 G.20
```

ONTMOETINGSWEEKEND OP 25, 26 EN 27 NOVEMBER 1983

TE RENDEUX - HAUT, NABIJ LA ROCHE IN DE ARDENNEN

Ter gelegenheid van haar stichtingsverjaardag nodigt de Belgische sectie van de Acorn Computer Club Nederland alle leden uit voor een vriendschappelijke ontmoeting.

Op het programma:

- een reuze Acorn Atom WORKSHOP
- een fantastische Acorn Atom wedstrijd
- een buitengewoon SYMPOSIUM rond de Acorn Atom.

Ook gezinsleden zijn welkom. Voor hen zal een apart programma worden opgesteld.

Dit ontmoetingsweekend staat geheel in het teken van de vraag: "Wat doen wij met onze Atom?"
De bedoeling is dat de diverse regionale Acorn-clubs (dat zijn er dus 14) zich in dit ontmoetingsweekend presenteren en manifesteren naar elkaar, zodat over en weer een beter inzicht ontstaat in die zaken die bij iédere bepaalde club het meest in de belangstelling staan, en welke resultaten er in dit opzicht reeds te melden zijn. Vandaar de WORKSHOP, waarbij elke deelnememende regionale club een eigen STAND mag inrichten volgens haar eigen smaak en met haar eigen stokpaardjes.
Ook aan de WEDSTRIJD wordt per regionale club deelgenomen:

➡ Heeft UW regio al een programma op het oog waarmee ze wil deelnemen????????

Zoniet, dan wordt het tijd om de koppen eens bij elkaar te steken. 'n Goed onderwerp voor de eerstvolgende bijeenkomst!

In het aansluitende SYMPOSIUM worden vervolgens de eerste conclusies getrokken uit deze confrontatie van regio's met hun resp. specialismen en visies.

Tenslotte: de deelname is beperkt door het beschikbare aantal kamers (80).

➡ Willen daarom de sectie-coördinatoren die dat nog niet deden zo spoedig mogelijk hun regionale deelnemerslijst insturen??

I N   D I T   N U M M E R

<u>In andere Acorn Clubbladen:</u>

<u>Acorn Nieuws Noord, mei '83.</u>

Uitbreiding funktietoetsen van CHARON/ (Netjes) uit een lus springen/ Video display Generator Extensie, met FLASHING CURSOR en meer van dat moois/ Omzetting X-Y-coördinaten naar Poolcoördinaten/ Relatie tussen (cassette-)bandteller en (Acorn-)bloknummers/ Berekenen van Veeltermen/ Snelle Sinus en Cosinus, door INTEGERE aanpak.

<u>Acorn Nieuws Noord, juni '83.</u>

Sorteerprogramma(naar alfabetische volgorde)/ Surrounder(Boter, Kaas en Eieren op de Atom)/ Regel van Cramer, bij stelsel van 2 of 3 vergelijkingen/ Zeer globale berekening van de zonne-declinatie.

<u>De CURSOR, 4/'83.</u>

Tijdmeting met optische sensors, een soort stopwatch/ PERT, een planningsmethode/ Simulatie van een golf in een koord.

<u>Acorn NEWSletter (België) 1/'83.</u>

Verwijderen van Assembler routines uit BASIC-programma's (lijkt nogal op artikel van Fred Danckaarts, pag.40)/ Uitleg van de EPROM-programmer/ Toelichting op de A/D-converter van AN 1/'83/ Gebruik van "ON ERROR" in loops/ Atom calculator pad, 'n numeriek toetsenbordje erbij/ Zelf een schaakprogramma maken?/ EPROMS in het gebruik, uit het HCC-blad/ Programma "Sinusjes" zonder TOOLBOX.

<u>Acorntjesbrood, 1.</u>

Project: 80-koloms video-display unit/ Opmerkingen met betrekking tot de 16K-geheugenkaart: Write Protect, Battery Backup, tweede 16K-kaart, kaart uitschakelen(disabelen).

(Oef! het gaat er juist op! red.)

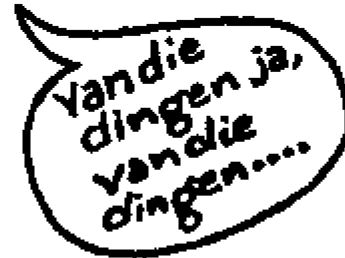
ERROR in "BENCHMARKING"

In het artikel "BENCHMARKING" van J.R. Marks in AN 3 ('83) is helaas door de auteur een nulletje over het hoofd gezien. Hierdoor verandert evenwel de conclusie; nogal drastisch eigenlijk.

Tot zover het slechte nieuws.

En dan nu het goede nieuws: de opgegeven tijd voor de test nr. 8 (deze test hoofdzakelijk enkele Floating-Point-funkties, HONDERD maal achter elkaar en niet DUIZEND!)op de Acorn Atom is een factor 10 te hoog opgegeven.
Dit betekent dat op het onderdeel Floating Point onze Atom oprukt van de 15e plaats naar de 11e en in het Algemeen Klassement van de 13e plaats naar de 3e plaats!

Een fraai resultaat, nietwaar?



BEVINDINGEN VAN DE HARDWARE VERGADERING

a. Allerlei defecten in de Atom kunnen worden veroorzaakt door de niet zo goede ic-voetjes; vooral als de pootjes van de ic's bijgebogen zijn om ze gemakkelijker in te kunnen steken. Verdachte ic's uit voetje halen, pootjes in de normale stand buigen en ic weer terug steken.

b. Het is meer dan eens voorgekomen dat door een defecte voeding te hoge spanning op de Atom is gekomen, met alle gevolgen vandien. Een extra overspanningsbeveiliging (bijv. met thyristor) is aan te bevelen.

c. Men is tevreden over de disk-drive: het adresgebied EXXX van de schakelkaart dient dan geheel buiten werking gesteld te worden (pen 10 van decoder los). Overigens, ook hex 2000-2800 wordt voor de disk gebruikt.

d. Verscheidene mensen hebben gelijktijdig aan hardware ontwikkelingen gewerkt zonder van elkaars know-how te kunnen profiteren. Een betere informatie-uitwisseling is gewenst, met name over (nieuwe) club-hardware (documentatie schakelkaart !).

e. Een seriële 4 maal i/o RS 232 is bijna klaar.

f. Hoogste prioriteit heeft een telefoonmodem met standaard communicatieprotocol. Verder gaan de gedachten uit naar high-resolution VDU, lichtpen, apart numeriek toetsenbord, a/d converter, extra processor, andere adressering schakelkaart, bankselecting geheugens.

g. Er moet met nadruk op gewezen worden dat de ACC alleen zin heeft als de programma's van een Atom ook op de meeste andere Atom's kunnen draaien. Hardware aanpassingen dienen dus zoveel mogelijk software-compatibel te zijn met de standaard Atom.

ZO MAAR WAT GEDACHTEN VAN EEN NIEUW LID
++++++++++++++++

Zoals uit de titel blijkt, ben ik nog maar sinds korte tijd lid van de Acorn computer club en ik wil u graag vertellen dat ik diep onder de indruk ben van de enorme know how die in de club gebundeld is, zowel op het gebied van de hardware als van de software.

Toch vind ik het jammer dat ik vrijwel niets hoor en helemaal niets lees over de projecten waarmee ik mij (hoofdzakelijk) bezighoud.

Het eerste (en belangrijkste) project betreft het schrijven van LEERPROGRAMMA'S.
Dit soort programma's hebben de intentie - zoals het woord uiteraard al zegt - iemand iets te leren. Er moet dus sprake zijn van communicatie tussen een leerling en de computer.

De computer legt de leerlingen problemen voor, reageert op juiste en foutieve antwoorden, legt -indien nodig- iets nog eens nader uit, verwijst terug in de leerstof, gaat na of alles goed begrepen is, enz.

Het zal u duidelijk zijn dat de omvang van degelijke programma's aanzienlijk is. Het overdragen van een vrij beperkte hoeveelheid leerstof vraagt meestal al snel 10 à 15 kbyte.

Het schrijven van leerprogramma's is primair een didactische aangelegenheid; het resultaat in de computer brengen is het beheersen van een paar techniekjes, die men in een uurtje onder de knie heeft.

Enige jaren geleden schreef ik al dergelijke programma's voor UNIVAC- en IBM-computers, dus de "grote broers" van de ACORN, maar de laatste doet dit werk zeker zo goed.

Het enige bezwaar is dat de ACORN geen speciale programmeertaal heeft voor het schrijven van leerprogramma's. Ik was dus genoodzaakt eerst een stuursysteem te ontwikkelen om te voorkomen dat elk item gevolgd moest worden door een (groot) aantal "IF's".

Dankzij dit stuursysteem (dat zeker nog voor verbetering vatbaar is) kan ik volstaan met het intoetsen van een GOTO naar een regel met een label. De ACORN springt dan - aan de hand van de reacties van de leerlingen - snel en accuraat naar de juiste plaats in het programma.

Nu zou misschien de gedachte kunnen bestaan, dat het schrijven van leerprogramma's uitsluitend is voorbehouden aan didactisch geschoolden.

Ik ben echter van mening (en ik heb dat in de praktijk getoetst) dat mensen die:

- zaken goed op een rijtje kunnen zetten;
- zich kunnen inleven in de gedachtengang van een beginneling, en die
- hun gedachten in voor de leerlingen begrijpelijke bewoordingen kunnen formuleren,

met enige aanwijzingen leerprogramma's kunnen schrijven van een aanvaardbare kwaliteit.

Wanneer zou blijken dat voor deze materie belangstelling bestaat, wil ik daar nog wel eens op terug komen.

Tenslotte nog één opmerking over leerprogramma's: een programma dat een leerling in een kwartiertje kan doorwerken , vraagt zeker 40 à 50 uur voorbereiding alvorens het gebruiksklaar in de computer zit. Het schrijven heeft dan ook uitsluitend zin als grote aantallen leerlingen het programma kunnen gebruiken.

Tot zover mijn eerste project. Het tweede heeft daar totaal niets mee te maken.

Ik heb jarenlang gefilmd en ben enige jaren terug overgestapt op video. Dat heeft - zoals iedereen wel zal weten - zowel voor- als nadelen.

Een (voor mij) belangrijk nadeel is het niet kunnen maken van enkelbeeldopnamen. Het is niet mogelijk animatie- en tekenfilms te maken. Ook ontbreken daardoor een groot aantal truc-toepassingen, die met film wel mogelijk zijn.

Zo had ik b.v. de gewoonte erg veel werk te maken van titels. Het lag dus voor de hand dat ik naar mogelijkheden zocht om de ACORN dat te laten doen. De Acorn bleek inderdaad een ideale titelgenerator.

Toen ik echter trachtte het beeld van de Acorn op de video-recorder vast te leggen, werd dat een volslagen mislukking. De videorecorder "pikte" de 60 hz. van de Acorn niet.

Alleen bij versneld weergeven (picture search) was het beeld te zien omdat de recorder dan een soort eigen synchronisatie-signaal gebruikt.

Tot op dit moment behelp ik mij door met de videocamera de titels op te nemen van het scherm van een (zwart-wit) TV, maar mooi is het niet.

Ik wacht dan ook met spanning op de komst van een print die het uitgangssignaal van de Acorn laat voldoen aan de ECE-norm en waarmee (naar ik hoorde) ook het gebruik van kleuren mogelijk is.

Intussen heb ik al een hele serie titelprogramma's klaar.

Wilt u ook eens proberen?

```
  1 REM stijgende titels
  2 REM JBDEKKER
 1Ø P.Ø12; IN."HOEVEEL TITELS "'N; DIM TTN
 2Ø F.C= 1TO N; DIMB64; TTC=B; N.C
 3Ø F.C= 1TO N;@ =Ø
 4Ø P."VOER TITEL "C" IN "'
 5Ø IN. ØTTC; N.C
 6Ø P.Ø12;  ?#E1=Ø
 7Ø F.C= 1TO N; P.''''''''
 8Ø F.W=Ø TO 12Ø; WAIT; N.W
 9Ø P.''''''''''"     "ØTTC
1ØØ N.C; G.7Ø
```

Een geheel ander effekt krijgt u na het aanbrengen van de volgende wijzigingen:

```
 1 REM verspringende titels
7Ø F.C= 1TO N; CLEARØ; P.Ø3Ø
75 P.''''''''"     "ØTTC

          en DELETE regel 9Ø
```

Er is in deze programma's ruimte genoeg voor langere en dubbelregelige titels, zoals b.v.:

CAMERA:
PIET DE ZOEKER

DEKORS:
KAREL KWAST

REGIE:
JAN DE REGELAAR enz.

In deze programma's gaat het er om dat de gebruiker een willekeurig titels kan invoeren.

Uiteraard kunnen dat ook geïnverteerde titels zijn.

We gaan het programma nu nog een keer wijzigen en u zult zien dat het nu de enkelbeeldopnamen van de film zeer dicht benadert.

```
 1 REM grceiende titels
74 P.'''''''"    "
75 F.X=Ø TO LEN(TTC)
76 F.W=Ø TO 3ØØ; N.W
77 P.$TTC?X; ?#E1=Ø; N.X
```

De volgende stap die ik graag zou willen maken is het mengen van computer- en videobeelden. De titels kunnen dwars door het "live" beeld worden geprojecteerd. Met film deed ik dat ook al. Voor dat doel heb ik zelfs al een programma "ondertiteling" klaarliggen.

Mocht er ook voor dat onderwerp belangstelling bestaan, dan zet ik mij wel weer achter de schrijfmachine.

J.B.Dekker
Acorn computer club
Overijssel/Gelderland

Het opzetten van een programma(gedeelte)

Vaak wordt een programma opgebouwd met de creativiteit van het moment. Als er een oplossing is gevonden dan worden er weer een aantal regels bij geplakt. Een aantal nadelen van deze methode van programmeren zijn:

- De totaal structuur van het programma is hopeloos (spagettiestructuur). Hetgeen bijzonder vervelend kan zijn als er achter af nog iets verbeterd moet worden.
- Het programma is meestal niet erg efficient (zowel met geheugen als in de snelheid). Dit komt onderandere door dat er verschillende gedee-ltes niet goed op elkaar aansluiten.
- De kans op fouten is groot.

Het is daarom beter om wat meer gedisiplineerd te werk te gaan. Dat zou volgens de volgende werkwijze kunnen gebeuren. Voor ieder programma worden de volgende stappen afgewerkt:

- vooronderzoek
- logisch ontwerp
- programeren en testen.

In het vooronderzoek is het de bedoeling om alle informatie die van belang is voor het maken van het programma te verzamelen. Punten die bij het vooronderzoek aan de orde kunnen komen zijn onderandere:

- Welke of watvoor uitvoer moet het programma leveren.
- Welke of watvoor invoer is nodig om tot de gewenste uitvoer te kunnen komen.
- Welke bijzondere gebruikerseisen moeten aan het programma gesteld worden bv mag het programma stoppen als er inplaats van een getal een letter wordt ingevoerd.
- Is het mogelijk om het gewenste programma op de acorn te maken bv. is het geheugen groot genoeg, zijn de reken tijden niet te lang.
- Aansluitend op het vorige punt welke aanpassingen zij nodig om het programma mogelijk te maken bv. een geheugen kaart .
- Specifieke informatie nodig voor het programma bv. bij een wiskundig programma de benodigde formules.

Om een goede programma structuur te verkrijgen maken we een logisch ontwerp. Dit doen we in dit stuk volgens een Top-down methode (van hoog- naar laag niveau) mbv flowcarts.[x] Het gaat als volgt te werk. Eerst maken we van het totale programma een flowcart. Het is hierbij de bedoeling dat alleen de grote lijnen worden weergegeven. zie voorbeeld1. Vervolgens gaan we van ieder blok een nieuwe flowcart maken. Mocht hierbij blijken dat een flowcart te groot en/of te onoverzichtelijk wordt dan is het verstandig om een aantal activiteiten samen te nemen en deze verder uit te werken in een nieuwe flowcart. Ook kan blijken dat een bepaald gedeelte programma meerdere keren voorkomt dan is het het efficientst om dat bepaalde gedeelte in een aparte flowcart te zetten waar naar in de andere flowcarts naar verwezen kan worden. zie vb 2. Door op de beschreven manier te werken wordt een duidelijke blokken structuur verkregen.

Bij het omzetten naar programmacode is het de bedoeling dat de bij het logisch ontwerp opgebouwde structuur wordt gehandhaaft. Dit kan als volgt worden bereikt. Eerst wordt de flowcart van het totale programma (de hoofdroutine) in code gezet. De verder uitgewerkte blokken worden daarbij gezien als subroutines (gosub's). De subroutines worden vervolgens in volgorde van aanroep achter de hoofdroutine geplaatst. De gedeeltes die meerdere keren voorkomen worden als subroutine achter in het programma geplaatst. Voorts zijn er nog een aantal punten die kunnen bijdragen aan de doorzichtigheid van het programma : - maak altijd een kop met mbv rem-statements waarin staat wie het programma gemaakt heeft, welke datum de laatste verbetering is aangebracht en hoe het programma heet.

- gebruik labels voor het springen naar subrouines
- dimensioneer (dim) grote arrays en strings altijd aan het begin van het programma (voor de hoofdroutine).
- plaats niet onnodig veel commandos op één regel.

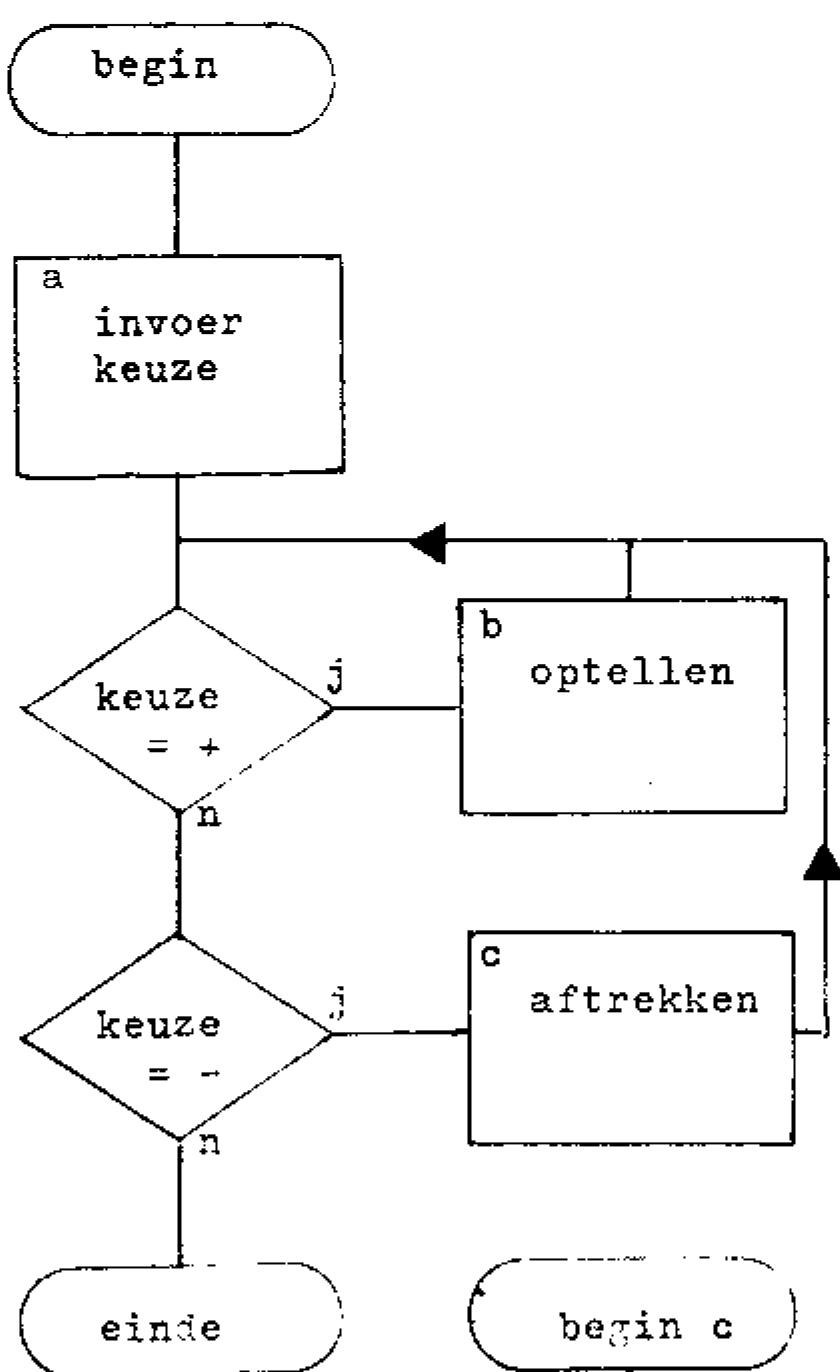
x het werken met flowcarts [illegible] besproken in '[illegible] theory and practice' ([illegible] handleiding).

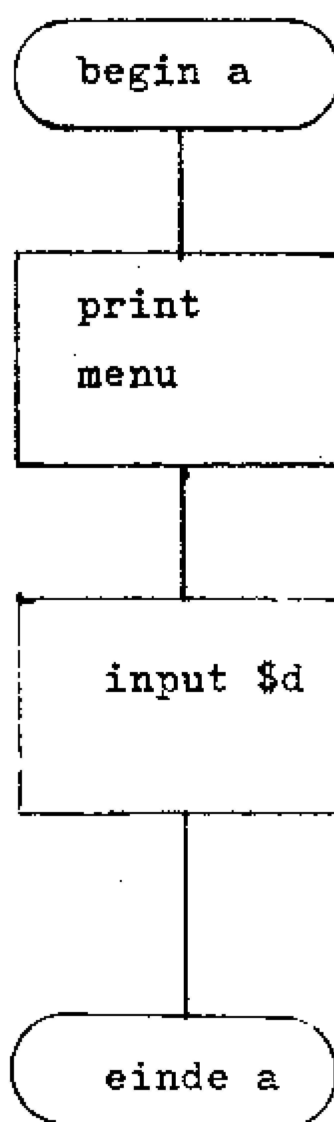
Om zeker te zijn dat er geen fouten meer in het programma aanwezig zijn is het verstandig voor het ingebruik nemen van het programma alle moglijke situaties te simuleren.

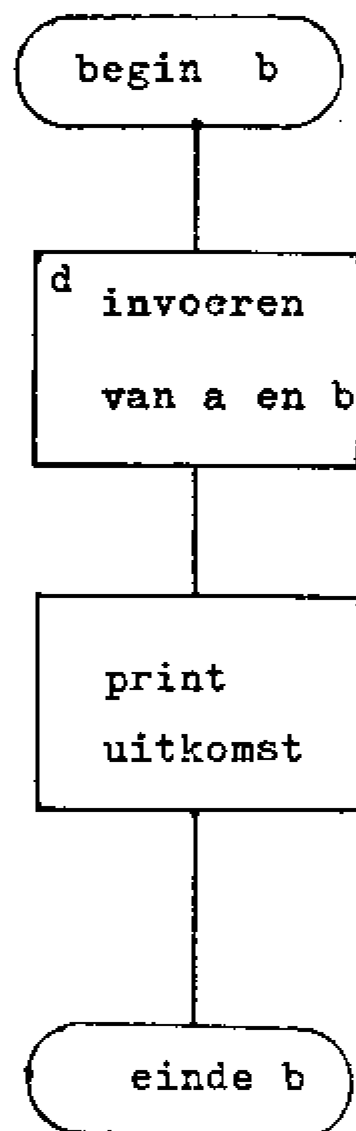
Nu een voorbeeld om een beeld te geven hoe het logisch ontwerp en het uiteindelijke programma er uit kunnen zien.

Erik Dukker, regio Delft.

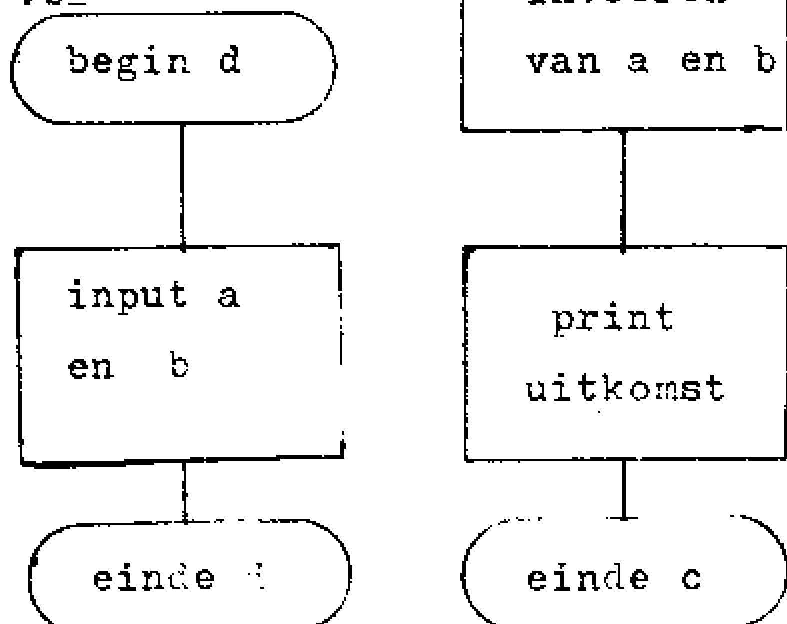
vb1







vb2



PROGRAMMACODE

```
 0 rem door : Erik Dukker 830705
10 dim d(65); gos.A
20 if $d="+"; gos.B; g.10
30 if $d="-"; gos.C; g.10
40 end
50A in."optellen +"''"aftrekken -"''"
    stoppen ret"''"keuze"$d;r.
60B gos.D;p.''"uitkomst = "(a+b)''
    ;r.
70C gos.D;p.''"uitkomst = "(a-b)'';r.
80D in."geef eerste getal"a,"geef tw
    eede getal"b;r.
```

Wanneer je de basisgegevens van programmatie en van de Basic-taal onder de knie hebt, wordt het interessant je programma's te trachten optimaal uit te schrijven, zodat ze niet alleen overzichtelijk en doorzichtig zijn maar ook minder kans maken op fouten.

Dit kan door meer "gestructureerd" te gaan programmeren, d.w.z. het programma een structuur geven door het volgen van een bepaalde werkmethode. Aldus wordt het programma ook gemakkelijker leesbaar en zal het tevens rapper op punt staan ; soms is het zelfs sneller in de uitvoering.

Principes "Gestructureerd programmeren"

1. Vraag je eerst af wat het programma moet doen i.p.v. te beginnen met de vraag "hoe" : het hoe komt later.

2. Maak vervolgens, op papier, een structuurschets in "pseudo code". De "pseudo-code" is een "pseudo-taal" die toelaat gemakkelijker over te gaan van gesproken taal naar de machinetaal. Die "pseudo-code" is zeer eenvoudig, soepel en er zijn slechts enkele "codewoorden" te onthouden.

3. Eenmaal de structuurschets af is, blijft het overschrijven naar de eigenlijke programmeertaal.

De "Pseudo-code" voor gestructureerd programmeren

Een structuurschets in "pseudo-code" maakt enkel en alleen gebruik van navolgende drie basisconfiguraties.

	Pseudo-code	overeenkomstig ordinogram
1 sequence (volgreeks)	S 1 S 2	[S 1] → [S 2]

	Pseudo-code	overeenkomstig ordinogram
2 IF...THEN... ELSE (als...dan.. zoniet)	IF condition THEN S 1 ELSE S 2 END IF	NO (else) / IF condition / YES(then) / S 2 / S 1
3 DO...WHILE (doe..inmiddels) d.w.z. zolang de conditie O.K. is	DO WHILE (UNTIL) condition do sequence END DO	condition / YES / NO / S

Opmerking : Wil men een sub-programma (="segment" in pseudo-code) dan gebruikt men het woord "INCLUDE" (tussenin-schakelen) ="GOSUB"

bv.

```
PGM
 ¦  INCLUDE "behandeling"
END PGM
-----------
SEGMENT "behandeling"
 ¦ schrijf "ACORN"
END SEGMENT
```

Voorbeeld pseudo-code structuur.

Ik lees op cassette willekeurig opgenomen getallen tot ik een "Ø" (zero) tegenkom.
Tijdens het lezen schrijf ik de getallen op in tabelvorm.
Daarna zoek ik de grootste waarde op - die noem ik "max." - en teken ze op met de respectievelijke index-teller-stand.

Opmerking

PROGRAM

```
- Dimensioneer tabel                    Op het einde zal de index (=
- indexstand = 1                        volgnummer) het totaal aantal
                                        elementen geven van de tabel
- lees opname
  DO...WHILE  opname ≠ 0
  |           index = index +1
  |           lees opname
  END DO

- max-index = 1                         ik initialiseer de index aan
                                        "max." op het eind van het
                                        programma.
- teller = 1
- max. = 0                              "max" ben ik niet verplicht te
                                        initialiseren
  DO ... WHILE  teller ≤ index          aldus ziet men of het eind van
  |                                     de tabel is bereikt
  |   IF  tabel (teller) > tabel
  |   |                   (max.index)
  |   |   THEN  max.index = teller
  |   |   (ELSE)                        ELSE staat tussen haakjes omdat
  |   |                                 het hier niets doet.
  |   END IF
  |   -teller = teller + 1
  END DO
- max = tabel (max.index)
- schrijf - max., max. index
```

END PROGRAM

Vertaling van de basis-configuratie naar BASIC

1. Sequence : in de opeenvolging van instructies

```
S 1
S 2
...
...
```

2. IF...THEN...ELSE :

```
100  IF  condition  GOTO  [A]
:     S 2
:    GOTO [B]
200 [A]  S 1
:
300 [B]
```

```
IF condition
|  THEN S 1
|  ELSE S 2
END IF
```

ofwel ---

```
100     IF  Condition    GOTO [A]
:            S 1
:            GOTO [B]
200 [A]      S 2
:
300 [B]
```

opm. : "$\overline{\text{condition}}$" is het omgekeerde van "condition"

- condition is A < B
- $\overline{\text{condition}}$ is A ≥ B

3. DO...WHILE

```
100 [A]  IF  Condition    GOTO [B]
:               S
200     GOTO [A]
:
300 [B]   vervolg van het programma
```

```
DO WHILE Condition
:   S
END DO
```

Opm. : DO...WHILE kan soms vertaald worden door een FOR...NEXT of zelfs een DO...UNTIL

Vertaling van het Voorbeeld in pseudo code structuur naar BASIC

```
10    DIM TT (100)              ik dimensioneer op 100 als ik
                                weet dat er minder dan 100 zullen
                                gelezen worden.
20    I = 1
30    A = FIN ""                hierdoor krijg  ik de melding
                                "PLAY TAPE"
40    GET A
100 [A] IF  A = Ø   GOTO  [B]
110   TT (I) = A
120   I = I + 1
130   GET A
140   GOTO [A]
200 [B] REM  Hier is het lezen van
             de cassette beëindigd
210   M = 1                     dit is de max.-index
220   N = Ø                     dit is MAX die ik moet initialiseren
230   FOR X = 1 to I
240   IF TT (x) > TT (M) then
      M = X
250   NEXT  X
260   N = TT (M)
270   PRINT "Het MAXIMUM is" N
280   PRINT "De INDEX van MAX is" M
290   END
```

Opmerking :

Altijd programma's trachten te maken met zo weinig mogelijk GOTO's ; niet bang zijn sub-programma's te maken. Alleen het programma uitschrijven na er goed over nagedacht te hebben en de structuur ervan eerst op papier zetten.

ASSEMBLER CURSUS DEEL 7

Deze aflevering gaat over de tot nu toe nog niet behandelde adresseer methode en instrukties. Behalve de interupt instrukties zijn na deze aflevering alle 6502 instrukties behandeld. In de volgende aflevering zullen de interupts worden behandeld (zowel hardware als ook de software interupt.)

DE PRE-INDEXED INDIREKT

Allereerst de nog overgebleven adresseervorm: pre indexed indirekt. De schrijfwijze voor deze, is LDA (#70,X). #70 is hier een voorbeeld. Het mag iedere zero page adres zijn. Wat gebeurt er: De waarde van het X register wordt bij de waarde van het zero page byte opgeteld, wat in het voorbeeld dus #70 is. Stel dat het X register met @#10 geladen is en #70 de waarde @#A0 heeft. X wordt nu bij #70 opgeteld. @#10 + @#A0 = @#B0. Dit @#B0 vormt het ADRES waar de instruktie nu gegevens vandaan haalt of naar toe brengt. Het lage deel van het adres staat op #B0 en het hoge deel va het adres op #B1. Om het te verduidelijken een voorbeeld als of het basic was.

```
10 H=?#70+X    (H IS TIJDELIJK HULPJE)
20 P=?H
30 P=P+?(H+1)*#100
```

Dit is een vrij ingewikkelde manier van adresseren, vandaar dat deze ook pas na 7 afleveringen wordt behandeld. Deze adresseervorm wordt niet echt vaak gebruikt. (een troost voor degene die het na 3 maal gelezen te hebben nog niet begrijpen.)
Een voorbeeld:

```
10 REM PRE INDEXED INDIREKT
20 P=#2800
30 ?#B0=0;?#B1=#80;?#70=#A0
40 [;LDA @#2A
50 LDX @#10
60 STA (#70,X)
70 RTS;]
80 LINK#2800;END
```

Dit programma zet een sterretje links boven in beeld. De BASIC equivalent zou zijn:

```
10 REM PRE INDEXED INDIREKT BASIC
20 ?#B0=00;?#B1=#80;?#70=#A0
30 X=#10
40 A=#2A
50 H=?#70+X
60 P=?H
70 P=P+?(H+1)*#100
80 ?P=A
90 END
```

Pre indexed indirekt mag alleen met het X register. Met het Y register kan dus niet.

De instruktie BIT

De BIT instruktie is een variant op de AND funktie. Bij een AND instruktie wordt de accu ge 'AND' met een getal en het resultaat komt in de accu te staan. Bij een bit funktie wordt ook ge AND doch tussen de WAARDE van de accu en een ADRES. Het enige wat gebeurd is dat het status register wordt bijgewerkt. In de accu veranderd niets. Met een BIT funktie kan b.v. gemakkelijk worden getest of een bepaald bit uit een adres hoog of laag is.
VOORBEELD:
P=#2800:{:LDA@#04:BIT #2800:JMP #2800
?#2800 heeft nu de waarde #A9, want dat is de machine code voor LDA @**. Bij de BIT test zal nu het volgende gebeuren:

```
  ACCU 0000 0100
?#2800 1010 1001
       --------- &
       0000 0000
           ↑
```

Het resultaat is 0, dus de Z flag is hoog. Omdat deze hoog is, is het dus absoluut zeker dat bit 2 van ?#2800 laag is. Als de accu geladen wordt met een andere waarde dan kan er b.v. een ander bit getest worden.
Met een BIT test gebeurt nog iets speciaals. Als het meest linkse bit van (adres) hoog is, dan wordt het N bit uit het status register hoog (zoals bij iedere bewerking die groter is dan #7F). Het speciale nu is, dat als bit 6 (de op één na meest linkse) hoog is, de overflow flag V geset wordt, en is bit 6 laag dan wordt of is de overflow flag laag. Dit is gemakkelijk te proberen! de repeat toets van onze ACORN zit vast aan bit 6 van #B002. Als met de MONITOR door het volgende programmatje ge-traced wordt, dan ziet u als u de rept toets indrukt, dat de V flag geset wordt. P=#2800:{:BIT#B002:JMP#2800

De BMI en BPL

Daarnet is al gesproken over het N bit. Dit bit wordt geset zodra een bewerking een uitkomst groter dan #7F heeft. Met de BMI wordt gesprongen als de N flag 1 is. Met de BPL wordt gesprongen als de N flag 0 is. voorbeeld:

```
10 REM BMI EN BPL TEST
20 DIM LL0
30 P=#2800:[
40 LDY@#7F:LDA@CH"+"
50:LL0 STA #8000,Y
60 DEY
70 BPL LL0
80 RTS:]
90 LINK#2800:END
```

De BVC en BVS

Zo'n zelfde verhaal geldt ook voor het V bit. De V staat voor Overflow flag. BVC Branch on oVerflow Clear en BVS Branch on oVerflow Set. voorbeeld:

```
10 REM OVERFLOW DEMO
20 P=#2800:[
```

```
30 BIT #B002
40 BVC #2800
50 JSR#FD1A     BLEEP ROUTINE
60 JMP #2800
70 ];LINK#2800
```

Dit laatste voorbeeld brengt nog een klein probleem aan het licht. Om dit programma te stoppen is een 'BREAK' nodig, want de ESCAPE toets doet niets. In BASIC wordt voordat een nieuw statement wordt geinterpreteerd, eerst gekeken of de escape toest werd ingedrukt. Zo ja, dan springt hij naar #C2CF en zo nee, dan gaat hij vrolijk verder met interpreteren. Omdat in het voorbeeld niet gekeken wordt naar de escape toets, heeft de esc. toets ook geen effekt. Toch is dit er eenvoudig in aan te brengen, de esc. toets zit namelijk aan het vijfde bit van #B001. Als deze wordt ingedrukt, dan is deze laag. De toets kan dus als volgt worden gesignaleerd:

```
        LDA #B001
        AND @#20      TEST OF VIJFDE BIT HOOG IS
        BNE VERVOLG   ZO JA, VERVOLG
        JMP #C2CF     ZO NEE, DAN NAAR ESCAPE
VERVOLG *****
```

Het voorbeeld zou er nu als volgt uit kunnen zien:

```
10 REM ESCAPE FUNKTIE
20 DIM LL2;LL1=-1
25 FOR N=0 TO 1
30 P=#2800;[
40:LL0 LDA  #B001
50     AND @#20
60     BNE   LL1
70     JMP  #C2CF
80:LL1 BIT  #B002
90     BVC   LL0
100    JSR  #FD1A   BLEEP
110    JMP   LL0
120 ]:NEXT N
130 LINK LL0
140 END
```

Het kan zo, maar het kan ook met een ROM routine. In de BASIC ROM zit namelijk ook al zoiets. Als gesprongen wordt naar #C504 (met een JSR) dan springt deze zelf naar #C2CF als de esc. is ingedrukt en als deze niet is ingedrukt dan keert deze via een RTS weer terug. voorbeeld:

```
10 REM ESCAPE MET ROM ROUTINE
20 P=#2800;[
30 JSR  #C504
40 LDA @#50
50 JSR  #FFF4   PRINT ACCU ALS ASCII OP BEELD
60 JMP  #2800
70 ];LINK#2800;END
```

De NOP instruktie

Deze instruktie (No Operation) wacht gedurende 2 clock pulsen, verhoogt de programm counter met 1 en vervolgt met de volgende instruktie. Het is een 1 byte instruktie en bewerkt geen registers. De NOP kan worden gebruikt in programma's waar de precieze tijd een eis is, b.v. frequenties e.d. of ter opvulling als een bepaalde instruktie moet vervallen. Deze kan dan worden vervangen door de NOP.

SAMENVATTEND:

*PRE INDEXED INDIREKT
*BIT AND DE ACCU EN EEN ADRES
*BMI EN BPL SPRINGEN ALS N BIT WEL OF NIET 1 IS
*BVC EN BVS " " V " " " " 1 "
*ESCAPEN MET ROM ROUTINE JSR#C504
*NOP

LEENDERT BIJNAGTE

```
 10 REM 3DIM. GRAFIC
 20 %Z=90
 30 CLEAR4
 40 FOR N=0 TO 180
 50   %S=2*RAD N
 60   %R=%Z*SIN(%S*2)
 70   X=%(%R*SIN%S+128)
 80   Y=%(%R*COS%S+96)
 90   IF N=0 THEN MOVE X,Y
100   DRAW X,Y
110   %R=%Z*COS(%S*2)
120   X=%(%R*COS%S+128)
130   Y=%(%R*SIN%S+96)
140   MOVE X,Y
150 NEXT N
160 END
```



EEN STUKJE PRAKTIJK VOOR DE ASSEMBLER CURSISTEN

Wanneer daar aanleiding toe is, wil ik de assembler cursus verduidelijken met een praktisch gedeelte. Het gaat hier om programma's die niet zozeer een effectieve waarde hebben doch meer het karakter van een leerzaam iets. Er is naar gestreefd een zo duidelijk mogelijk beeld te geven van de algemene Assembler programmeer stijl. Truukjes, grapjes en allerlei andere slimmigheden die een ervaren assembler programmeur weet, worden hier per se niet gebruikt. Er is gestreefd naar een effectieve schrijfwijze, maar niet altijd naar een zo kort mogelijke. Dit komt namelijk de duidelijkheid niet altijd ten goede. Verder sta ik open voor nieuwe suggesties die behandeld moeten worden in de assembler cursus, want de cursus loopt tegen het einde, nu alle instrukties behandeld zijn.
En dan nu het programma. Het is het omschrijven van een basic programma naar assembler. Het programma:

```
 10 REM CRC SIGNATURE TEST
 20 REM BRON: ATOM MANUAL BLZ. 93
 30 INPUT "ROM ADRES ",P
 40 C=0;Z=#FFFF;Y=#2D
 50 FOR Q=0 TO #FFF
 60   A=P?Q
 70   FOR B=1 TO 8
 80     C=C*2+A&1
 90     A=A/2
100     IF C>Z THEN C=C:Y;C=C&Z
110  NEXT B
120 NEXT Q
130 PRINT "SIGNATURE IS:#",&C
140 END
```

Het is altijd raadzaam een stroom diagram te maken van dit soort zaken. Zie de bijlage. Uit het stroom diagram weten we nu een aantal gegevens:
*we moeten steeds blokken van 4K nemen, dus 16 blokken van 255 bytes
 we nemen hiervoor de POST INDEXED INDIREKT
*delen door 2 is een LSR instruktie
*als C groter is dan #FFFF dan moet er ge 'exor'ed worden. C:Y

```
C=1111 1111 1111 1111
Y=0000 0000 0010 1101
  ------------------- EOR
  1111 1111 1101 0010
```

Hier uit volgt dat alleen het low-byte van belang is bij de exor
*we hebben 2 lussen:
 1 adres lus
 2 reken lus

De adres lus kan alvast genoteerd worden:

```
        LDA @#10
        STA  #70     TELLER VOOR 16 BLOKKEN VAN 255 BYTES
        LDY @0
```

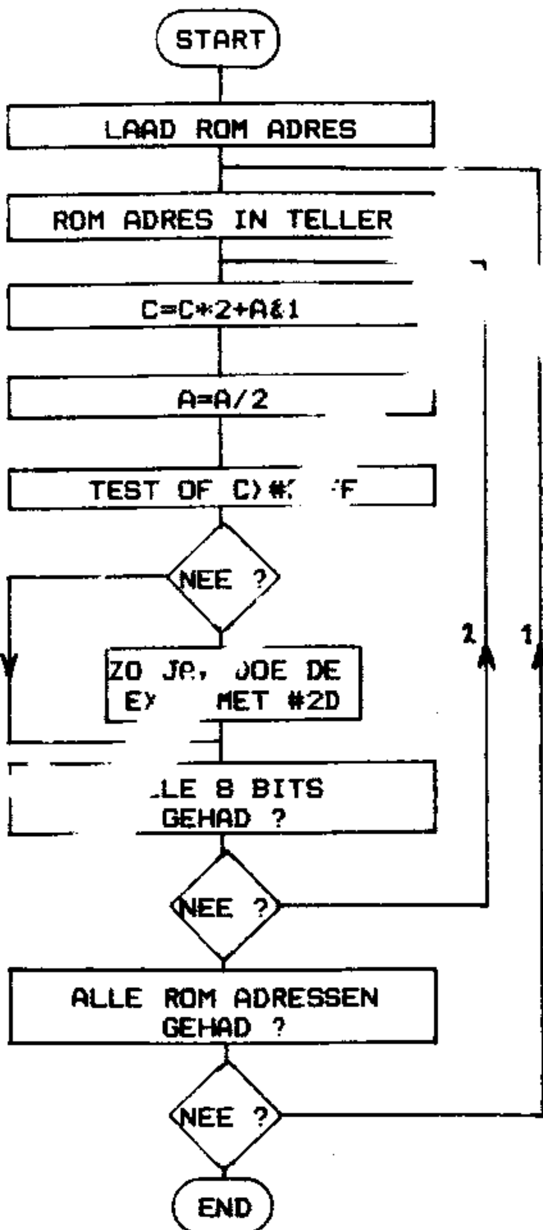
```
        (   #70
        STY  #7.. } SAMMEN MET #7B HET RESULTAAT REGISTER
        ST.  #7B
  NEXT  LD(  70),Y
        ...
        . .
        I JY
        BNE NEXT
        INC  #71
        DEC  #72  DEC 16 BLOK TELLER
        BNE NE..
        RTS
```

De buitenste lus uit het diagram is nu een feit. De 2 bytes vermenigvuldiging is een ROL instruktie. Als van het low byte een carry overdracht ontstaat kan deze, als direkt er achter aan het high byte wor.. geschoven, direkt worden door geschoven in het high byte. Als, door gevolg van de ROL instruktie van het high byte, weer een ..rry ontstaat, is het resultaat groter dan #FFFF en moet er dus ge exored worden. In regel 20 staat de A=A/2 opdracht. Als d.. wordt vervangen door een LSR instruktie, dan staat het bit wa.. eerts op bitplaats 0 stond, nu in de carry (na de LSR instrukt..e) Als direkt erna de vermenigvuldiging ROL ROL doen, dan wordt .. carry van de deling in low byte geschoven en de carry van low byte wordt in high byte geschoven en de eventuele carry van high byte wordt gebruikt om te testen of het resultaat groter is dan #FFFF. Het X register kan worden gebruikt voor de 8 aal 1 bit teler. De rekenlus zou er dus zo uit kunnen zien:

```
        LDX @#08       FOR B=1 TO 8
   :LL1 LSRA           ..=A/2
        ROL  #7A
        ROL  #7B       ..C*2+A&1
        BCC  LL2
        PHA            ..G A TIJDELIJK OP DE STACK
        LDA  #7A
        EOR  ?..2D
        STA  #7A
        PLA   HAAL A WEER VAN DE STACK
   :LL2 DEX            NEXT B
        BNE  LL1
```

Dit is het eigenlijke programma. De uitvoer is naar eigen fantasie. Het kan zoals het op de bijlage staat. Dit programma duurt 1.2 seconde en de basic versie duurde 6 minuten.
Probeer zelf ook eens dit soort kleine basic programma's eens te vertalen naar assembler, want op die manier komt de ervaring in het assembler programmeren. Een volgende keer zal ik het programma van Arie Marchal, wat iedere 16K RAMmer bij zijn print heeft gekregen, ver-machine-talen. Denk er eens over na en probeer het zelf vast op te lossen.

```
 5 REM AUTEUR: ARIE MARCHAL
10 @=2
20 FOR I=#3 TO #FF
30   A=I*256
40   IF ?A<>13 OR A?1 <>0 THEN GOTO 80
50   PRINT' &I "  "
60   J=3;DO
70        PRINT $A?J
75      UNTIL A?J=13 OR J=26
80 NEXT I
90 PRINT ';@=8;END
```




```
0REM CRC TEST
1REM BIJLAGE ASSEMBLER CURSUS
2DIM LL3
3FOR N=0 TO 3;LL(N)=-1 NEXT N
4FOR N=0 TO 1
5P=#2800; [
10        LDA @#10
20        STA #72
30        LDY @#00
40        STY #7A
50        STY #7B
60        STY #70
70:LL0 LDA (#70),Y
80        LDX @#08
90:LL1 LSR A
100       ROL #7A
110       ROL #7B
120       BCC LL2
130       PHA
140       LDA #7A
150       EOR @#2D
160       STA #7A
170       PLA
180:LL2 DEX
190       BNE LL1
200       INY
210       BNE LL0
220       INC #71
230       DEC #72
240       BNE LL0
250       RTS
260];NEXT N
270INPUT "ROM ADRES (BV. #F0)",P
280?#71=P
290LINK #2800
300C=?#7B*#100+?#7A
310PRINT "SIGNATURE IS :",&C'
320END
```

TOT DE VOLGENDE KEER

LEENDERT

-:---------**-----------:-

Oostkapelle, 4 juli 1983.

Aan: T. Otten
J.A. de Gravenlaan 17,
Zoeterwoude.

Hierbij deel ik u mee dat het telex-besturingsprogramma in het drukwerk-archief aanwezig is. Het is niet hetzelfde programma dat ik u destijds toegestuurd heb, maar het betreft een sterk verbeterde versie welke efficiënter, flexibeler, gebruiksvriendelijker en ook nog korter is.

Eigenschappen:
- koppelen met CTRL-B;(of PRINT $2);
- ontkoppelen met CTRL-C; (of PRINT $3);
- mogelijkheid voor A4-indeling;
- is geschikt voor zowel RAM als EPROM;
- gebruikt geen Zero Page adressen;
- evt. karakter-wijzigingen gemakkelijk aan te brengen;
- totale lengte is 354 bytes.

De aanwezige documentatie-set bestaat uit:

- source-listing van het programma;
- commentaar bij de assembler-listing;
- beschrijving voor het maken van de object file;
- gebruiks-aanwijzing;
- toelichting over het toegepaste principe;
- aanwijzingen voor alternatieve karakters;
- samenwerking met ROM's e.d. voor zover bekend;
- beschrijving en schema's van de benodigde hardware;
- literatuur opgave over dit onderwerp;

Het geheel omvat 8 A4-bladzijden en is verkrijgbaar onder de naam "telex-besturingsprogramma".
Ik hoop hiermee de club van dienst te zijn geweest.

Hoogachtend,

Afz: J. Jobse jr,
Duinweg 35,
Oostkapelle.

DRUKWERK

Dit is het tweede deel van het overzicht van het landelijke drukwerk-archief.Het eerste deel stond in acorn nieuws nr.2 '83 op bladzijde 43. In dit overzicht worden de categorieën ALGEMEEN en HANDLEIDINGEN besproken.De categorie OVERIGE PROGRAMMA'S is veel uitgebreider dan de 3 titels in het vorige overzicht maar het lijkt mij niet zinvol om dit verder in acorn nieuws te behandelen.Ook volgen hier nog enige aanvullingen op het hardware gebied.

ALGEMEEN	blz:	code:
- Newsletter Atom user group nr.1 '80	9	A 01
- id. nr.2 '80	12	A 02
- id. nr.3 '81	9	A 03
- id. nr.4 '81	11	A 04
- id. nr.5 '82	10	A 06
- Vergelijking VIC20/ATOM/PET/APPLE II	3	PC 01.82
- BBC microcomp. technische specifikatie	42	BBC 01
- BBC microcomputer,beschrijving (WW 03.82)	7	BBC 02
- Survey Atom software	3	YC 07.82b
- ROM routine adressen (Cxxx,Fxxx)	2	A 05
- CHIP test Acorn Atom	4	CH 06.81
- Review Acorn Atom	3	EC 10.81
HANDLEIDINGEN		
- Atom disc pack	20	M 01
- Atomstore	6	M 02
- ECD database	11	M 03
- Lisp theory & practice	74	M 04
- Forth	60	M 05
HARDWARE		
- BBC/Atom graphics digitiser	3	PC 03.83
- Schema TX als monitor (zie A.N. 2 '83)	2	H 06
- Telex als printer 3 interface schema's, het bestuuringsprogramma en een zeer goede beschrijving van dit programma.	8	H 07

Niet alle stukken zijn vrij verkrijgbaar.

Als U een artikel wilt bestellen informeer of het al in de regio aanwezig is. Is dit niet het geval dan kan het gewenste stuk bij mij worden besteld. (Als het mogelijk is via de "drukwerkman" van de regio). Bestelwijze: Alleen schriftelijk onder vermelding van titel(s) en codenummer(s). Gelijktijdig moeten de kosten worden voldaan. Dit kan door bijsluiting van een girobetaalkaart of een eurocheque. het bedrag kan ook worden overgemaakt op mijn girorekening nr. 4229155. (bedragen onder ƒ1,50 mogen in postzegels worden voldaan).

Kosten:	1- 3 kopieën (A4)	ƒ0,65	
	4- 7 kopieën	ƒ1,20	
	8-17 kopieën	ƒ1,80	plus ƒ0,15 per kopie
	18-45 kopieën	ƒ2,60	
	46-90 kopieën	ƒ4,75	

Een volledig overzicht van dit en een aantal regionale archieven kan bij mij worden verkregen. De kosten hiervan zijn ƒ2,00

Drukwerk archief Acorn computer club
p.a. Frank Benschop,
De Blauwe Wereld 49,
1398 EP Muiden.
tel: 02942-3650

Een KNIPPERROUTINE (uit: Acorn Nieuws Noord, mei 1983)

Met deze routine kan een gedeelte van het scherm knipperen (normaal-inverse-normaal etc.) totdat de "shift"toets ingedrukt wordt.

```
:LL1 LDY 0
     JSR ≠FB81     (voor langzamer knipperen: FB7A)
:LL2 LDA ≠816A     (begin adres van het knipperend vlak)
     EOR ≠80
     STA ≠816A     (                id.                )
     INY
     CPY ≠0C       (lengte van het knipperend vlak)
     BNE LL2
     LDA ≠8001
     AND ≠80       (AND ≠40 voor wachten op "CTRL" toets)
     BNE LL1
     RTS
```

Deze routine gebruik ik voor (opvallende) aankondigingen in een programma.

R. van Drunen.

HET BANDJESARCHIEF

Het is nu ong. een half jaar geleden dat ik het bandjesarchief van Mevr. Bruins heb overgenomen.Na enkele problemen in het begin is nu in de meeste regio's een copie van het totale archief aanwezig.Ik zal voor A.N.5 een lijst maken met de adressen van de dan bekende(hopelijk allemaal) regionale archief houders.
Verder ben ik bezig om het aantal banden (nu 10) met enkele uit te breiden.
Ik denk hierbij aan:

- een band met utilities(database,uitbreiding CTRL enz.)
- een band met spelletjes
- een band met programma's voor radio amateurs
- een band met wiskunde programma's.

Als er mensen zijn die denken dat ze interessante programma's hebben, die kunnen deze bij hun regionale archief afgeven.
Hieronder geef ik nog even het huidige archief.
Deze cassette's zijn bij de regio archieven te bestellen.
Weet u niet wie dit beheerd, neem dan contact op met het regio bestuur.(zie A.N. 1/'83).

Nico Stad (regio Noordholland)

BAND 1..........fruitautom.,yahtzee,cubicrow,breakout,art
BAND 2..........morse 1&2,rtty,fax
BAND 3..........div. disassemblers en monitoren
BAND 4..........alles uit atomic theory & practice
BAND 5..........schaakband
BAND 6..........vele utilities,soft-VDU,sorteren en printen
BAND 7..........div. programma's van Wim Ernst
BAND 8..........div. time data spelen
BAND 9..........monitor Roel Heuvel,schiphol,forth,dammen (gestapeld geheugen minimaal vereist)
BAND 1Ø.........vele hobbiescoop programma's (gestapeld geheugen minimaal vereist)

2 PROGRAMMA'S EN HUN COPIEEN 2

Iedereen die fabrieksprogramma"s heeft en deze regelmatig gebruikt ergert zich elke keer weer aan het feit dat het laden zolang duurt.Elk ander programma kun je makkelijk op 12ØØ BAUD overzetten,maar de meeste gekochte programma's zijn beschermd.
Hoe worden deze programma"s nu beschermd:

Bij het SAVEN van in programma in de fabriek wordt in de programma naam een contrôle-code meegegeven.Dat zijn vaak de volgende twee:

-CTRL-C (P.$3) = printer uit
-CTRL-U (P.$21) = beeldscherm uit

Deze worden dan voor of achter de naam geplaatst en via een hulpprogramma wordt dan het hoofdprogramma opgestart.
De naam en de CTRL-codes zijn nog wel via het startprogramma te achterhalen, maar de adressen waar het programma moet komen en het startadres dat kennen we niet.
Ik zat zelf met een aantal programma's ook met dit probleem en dacht bij mijzelf deze gegevens moeten toch ergens in de ACORN geplaatst worden.

Er is in ons speeltje maar één plaats waar ik ze zou kunnen vinden en dat moet de ZERO-PAGE zijn.

En ja hoor daar waren ze te vinden!!!!

Deze adressen worden door K. v. Houten in zijn programma COPY (A.N. 3/'83 blz.59) ook gebruikt.
Als we na *CAT in de Z.P. gaan kijken dan vinden we daar alles dat we nodig hebben.
Als voorbeeld neem ik "CHESS" van P.P. dit omvat drie delen.

1- CHESS het startprogramma
2-?????? het hoofdprgramma
3- DATA het dataprogramma

In de Z.P. kijken we naar de volgende adressen:

- #D4 & #D5	hier staat het BLOCKadres	#28ØØ
- #D6 & #D7	hier staat het LINKadres	#378A
- #D8	de BLOCKlengte	#FF
- #D9	het BLOCKnummer	#ØØØØ
- #ED	van hieraf de naam	" AJM"

Op de twee spaties voor de naam staan de contrôlecodes 3 & 21.
Nu kunnen we met 12ØØ BAUD het programma weer saven.

Veel sterkte ermee,
Nico Stad

Beste Ton.

Hierbij stuur ik je het herziene overzicht van block zero.
Via deze weg wil ik ook G.B. Poot bedanken voor zijn aanvullingen en suggesties. Hij merkte terecht op dat ik niet reageerde naar aanleiding van zijn stukje in AN3, maar dat ik schreef op het artikel Stack & Queue van Frans (blz 19-21).
Sorry dat ik dat niet duidelijk vermeld had.
Als er mensen zijn in het land die aanvullingen op dit overzicht hebben dan zou ik die erg graag toegezonden krijgen.
Met deze aanvullingen is het dan mogelijk om t.z.t. een nieuw overzicht te publiceren.

Een programma om de FOR-NEXT stack te bekijken is:

```
 10FOR I=#240 TO #2B8
 20   ?I=0
 30NEXT
 40FOR A=1 TO #11121314 STEP #15161718
 50 FOR B=1 TO #21222324 STEP #25262728
 60  FOR C=1 TO #31323334 STEP #35363738
 70   FOR D=1 TO #41424344 STEP #45464748
 80    FOR E=1 TO #51525354 STEP #55565758
 90     FOR F=1 TO #61626364 STEP #65666768
100      FOR G=1 TO #71727374 STEP #75767778
110       FOR H=1 TO #81828384 STEP #85868788
120        FOR I=1 TO #91929394 STEP #95969798
130         FOR J=1 TO #A1A2A3A4 STEP #A5A6A7A8
140          FOR K=1 TO #B1B2B3B4 STEP #B5B6B7B8B
150           PRINT"ESCAPE PLEASE"
160          NEXT
170         NEXT
180        NEXT
190       NEXT
200      NEXT
210     NEXT
220    NEXT
230   NEXT
240  NEXT
250 NEXT
260NEXT
270END
```

Een XDUMP van #240 tot #2B8 levert de grenzen van de stacks.

Met vriendelijke groeten
Theo van Rijn.

Afz: Th. van Rijn.
Bankastraat 56
2585 EP Den Haag.

BLOCK 0

adres hex	inhoud
0	error nummer
1,2	lijn nummer dat geinterpreteerd wordt
3	pointer naar het character dat onderzocht wordt
4	basic stackpointer (x)
5,6	startadres statement dat geinterpreteerd wordt.
7	count cq kolomcounter.
8,C	random nummer seed
D,E	top
F	werkruimte.
10,11	adres van basic ERROR handler, normaal C9E7
12	text space pointer
13	aantal actieve DO-loops.
14	aantal actieve GOSUB 's.
15	aantal actieve FOR-NEXT loops.
16-22	basicstack laagste byte
23,24	pointer voor DIM
25,33	basicstack, een na laagste byte
34,42	basicstack, een na hoogste byte
43,51	basicstack, hoogste byte
52,59	basic werkruimte(zie bv jsr#C3C8)
5A,5B	x-coordinaat plotroutine.
5C,5D	y-coordinaat plotroutine.
5E	0=move,1=set,2=invert,overige clear (plotroutine)
5F,60	werkruimte plotroutine.
61,6F	basic werkruimte.
70,7F	floating point werkruimte.
80,AF	vrij;in gebruik bij AXR1, Josbox, monitor en FPToolbox. (Zie gedetailleerd overzicht van dit gebied op pagina)
B0,BF	DOS werkruimte
C0,DB	COS werkruimte.
DC	check som lees en schrijven van/naar cassetteband.
DD	FLOAD indicatie:bit 7=1 negeert bloknummers.
DE,DF	VDU-ram adres begin van de regel waar de cursor staat DE bevat 00,20,40 enz. ;DF bevat 80 of 81.
E0	cursor positie in VDU-regel bevat 00 tot 3F, na P.$21 (stop scherm) 80 tot BF
E1	vorm van de cursor:0=weg,#80=wit blokje.
E2,E5	werkruimte.
E6	aantal regels die nog op het scherm kunnen als VDU mode; bit 7=1 pagemode off.
E7	soort characters dat gevormd wordt.?231=0 terug naar normaal.
E8,E9	werkruimte.
EA	MON als nul, NOMON als negatief, als positief wel PLAY en REWIND TAPE, maar niet RECORD TAPE.
EB,FD	COS werkruimte.
FE	character dat niet naar de printer gezonden wordt(normaal 0A,linefeed)
FF	werkruimte.

100,13F input lijnbuffer.
140,17F string input buffer.
180,1FF 6502 stack(neerwaards)
200,201 NMI routine adres;niet geinisaliseerd door BASIC of COS.
202,203 BRK routine adres,normaal C9D8
204,205 IRQ routine adres;normaal A000.
206,207 lijninterpreteer routine adres,normaal F8EF.
208,209 write charakter routine adres;normaal FE52
20A,20B lees charakter routine adres;normaal FE94
20C,20D load file routine adres;normaal F96E
20E,20F save file routine adres;normaal FAE5
210,211 RDRVEC routine adres;normaal C2AC (BRK)
212,213 STRVEC routine adres;normaal C2AC (BRK).
214,215 get byte from tape rtne adres;normaal FBEE.
216,217 put byte to tape rtne adres;normaal FC7C.
218,219 print tape boodschap routine adres;normaal FC38.
21A,21B shut file rtne adres;normaal C278 (RTS).
21C,23F vrij;WHILE;ENDWHILE stack (PPT).
240,24A naam FOR-NEXT variabele stack, 11 groot.
24B,255 FOR-N stack: stap grootte laagste byte.
256,260 FOR-N stack:stap grootte 2e byte.
261,26B FOR-N stack;stap grootte 3e byte.
26C,276 FOR-N stack;stap goootte hoogste byte.
277,281 FOR-N stack;TO waarde laagste byte.
282,28C FOR-N stack;TO waarde 2e byte.
28D,297 FOR-N stack;TO waarde 3e byte.
298,2A2 FOR-N stack;TO waarde hoogste byte.
2A3,2AD FOR-N stack;return adres lage byte.
2AE,2B8 FOR-N stack;return adres hoge byte.
2B9,2C3 DO loop stack;return adres lage byte, 11 groot.
2C4,2CE DO loop stack;return adres hoge byte.
2CF,2DC GOSUB stack;return adres lage byte, 14 groot.
2DD,2EA GOSUB stack:return adres hoge byte
2EB,320 ARRAY POINTERS:

@@	AA	BB	CC	DD	EE	FF	GG	HH	II	JJ	KK	LL	MM	NN	OO	PP	QQ	RR	SS	TT	UU	VV	WW	XX	YY	ZZ
2EB	2EC	2ED	2EE	2EF	2F0	2F1	2F2	2F3	2F4	2F5	2F6	2F7	2F8	2F9	2FA	2FB	2FC	2FD	2FE	2FF	300	301	302	303	304	305
306	307	308	309	30A	30B	30C	30D	30E	30F	310	311	312	313	314	315	316	317	318	319	31A	31B	31C	31D	31E	31F	320

321,3BC INTEGER VARIABELE:

@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
321	322	323	324	325	326	327	328	329	32A	32B	32C	32D	32E	32F	330	331	332	333	334	335	336	337	338	339	33A	33B
33C	33D	33E	33F	340	341	342	343	344	345	346	347	348	349	34A	34B	34C	34D	34E	34F	350	351	352	353	354	355	356
357	358	359	35A	35B	35C	35D	35E	35F	360	361	362	363	364	365	366	367	368	369	36A	36B	36C	36D	36E	36F	370	371
372	373	374	375	376	377	378	379	37A	37B	37C	37D	37E	37F	380	381	382	383	384	385	386	387	388	389	38A	38B	38C

38D,3C0 ADRES VAN LABEL:

-	38D	38F	391	393	395	397	399	39B	39D	39F	3A1	3A3	3A5	3A7	3A9	3AB	3AD	3AF	3B1	3B3	3B5	3B7	3B9	3BB	3BD	3BF
-	38E	390	392	394	396	398	39A	39C	39E	3A0	3A2	3A4	3A6	3A8	3AA	3AC	3AE	3B0	3B2	3B4	3B6	3B8	3BA	3BC	3BE	3C0

3C1,3FC BASIC werkruimte.
3FD getal m.b.t. COLOUR.
3FE,3FF adres van point plot routine:
F6E2 voor mode 0
F738 voor mode 1
F754 voor mode 2
F76D voor mode 3
F7AA voor mode 4.

RUIMTE

in gebruik bij de Toolboxen van Program Power(PPT), ECD(AXR1), en Acorn Computer Club (Josbox).

adres hex	inhoud
80,85	STEP en TRACE werkruimte (PPT).
86	PPT: 1=AUTOmatisch lijnnummering,0=geen AUTO.
87,88	regelnummer in AUTO mode (PPT).
89,8A	increment in AUTO mode (PPT).
8B,8F	PPT werkruimte.
90,97	AXR1 werkruimte,
	PPT werkruimte,
	monitor werkruimte.
98,99	DATA pointer AXR1,
	PPT werkruimte,
	monitor werkruimte.
9A,9F	ECD-DOS werkruimte.
	monitor werkruimte.
A0,A5	monitor werkruimte.
A6	aantal actieve WHILE-loops PPT.
	monitor werkruimte.
A7	laatste XIF conditie PPT.
	monitor werkruimte.
A8,A9	regelnummer met de laatste DATA(PPT)
	monitor werkruimte.
AA,AB	vrij
AC,AD	DATA pointer (PPT).
AE.	-1 als PPT commando.
AF	PPT werkruimte.



Wie vult dit aan? (red.)

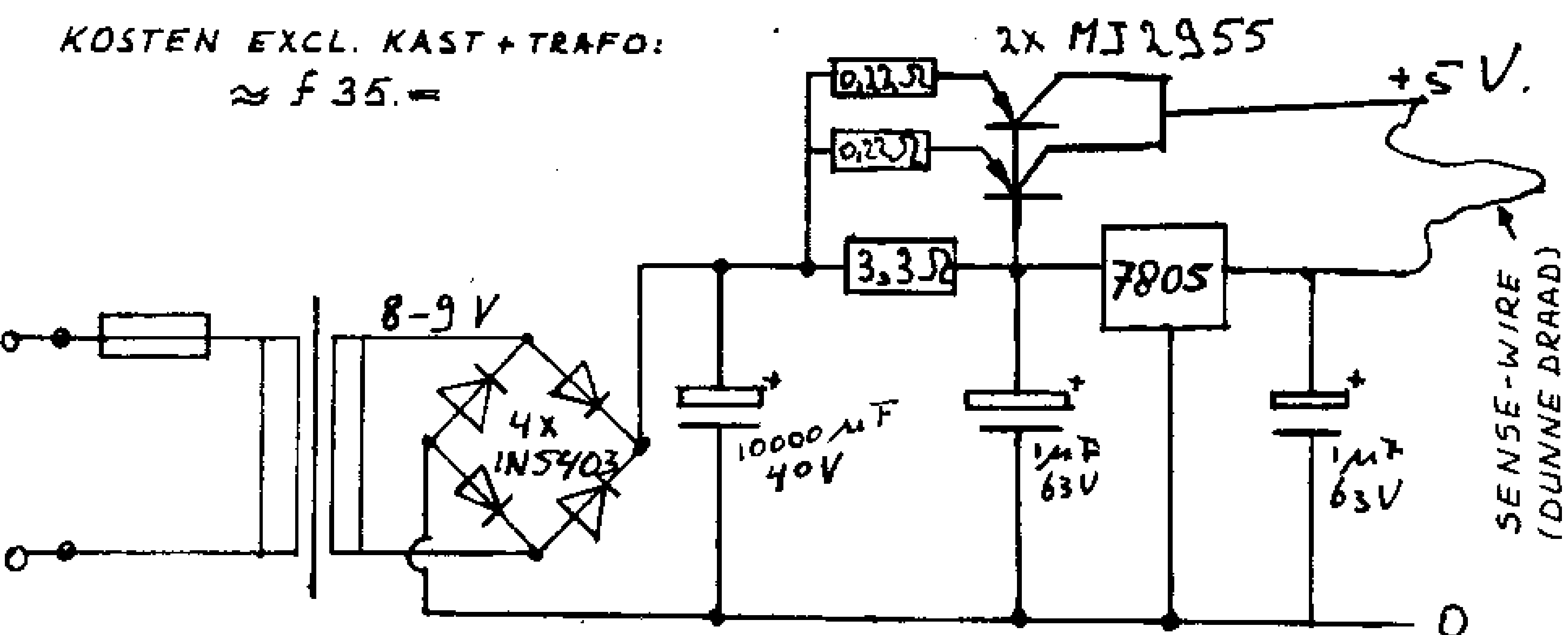
PPT = Program Power Toolbox.
AXR1 = Josbox.

Het weten waard.

Het handboek vermeldt:
#2800-#2886 Ruimte voor FL-point variabelen %@-%Z
Maar wist U dit al:
#2887-#28A1 Low-byte van FL-point array's
#28A2-#28BC High-byte van FL-point array's
#28BD-#28FF Vrij !!!

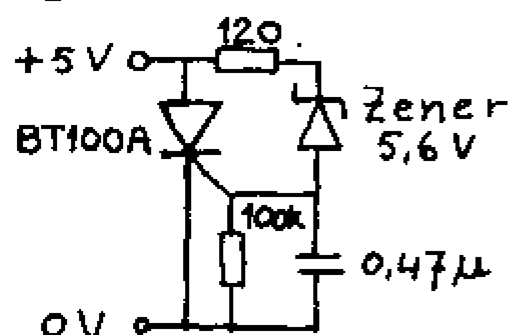
Er zijn al diverse artikelen en ontwerpen over en van voedingen in het afgelopen jaar in ons clubblad verschenen.
Daar zou ik graag nog een voedingsontwerp aan toe willen voegen.
Het is een ontwerp voor een smalle beurs en voor een gegarandeerde 5W op de print.
Dit komt door de derde draad(een soort zenuw),die op diverse plaatsen op de print op de 5 Volts lijnen wordt aangesloten.
Als regulator wordt een 78Ø5 gebruikt en de twee MJ2955 zorgen er voor dat er door de 78Ø5 niet meer dan 2ØØ mA te slikken krijgt.Het is mogelijk om met dit schema een voeding tot ongeveer 20 A te maken.(als de transformator maar zwaar genoeg is)
Dit ontwerp is afkomstig van Erik de Koning.Met Dank.

Nico Stad



Noot van de redactie:

Bovenstaande schakeling, die uitmunt door eenvoud, zou nog wat veiliger kunnen voor onze Atom, zodat ingeval van storing de Atom heel blijft. Als de spanning nl. onverhoopt belangrijk boven 5V stijgt, dan gaat er in de Atom in één klap erg veel kapot. Een simpele overspanningsbeveiliging (zie nevenstaande schakeling uit Acorn Nieuws 2, '82) voorkomt dit:



Eigenlijk zou er dan ook nog een kortsluitbeveiliging moeten zijn, bijv. een zekering van voldoende sterkte tussen de + van de elco van 10.000 µF en de rest van de schakeling. Dan blijft de voeding zélf ook zoveel mogelijk héél.

assembler programmering

Het is mij op de eerste bijeenkomst van de regio Den Haag opgevallen, hoe weinig mensen iets wisten van het direct machinetaal programmeren op de atom. Daarom zal ik in dit artikel zo kort en bondig mogelijk dit proberen uit te leggen aan de hand van voorbeelden.

Stel u wilt onderstaand programma door de atom laten uitvoeren.

```
10 DIM P-1
20 {
30 JSR#FFE6;STA#322
40 RTS
50 }
60 END
```

RUN

```
20 2933
30 2933 20 E6 FF JSR#FFE6
30 2936 8D 22 03 STA#322
40 2939 60       RTS
```

Dit programma neemt 51 byte's in beslag. Om dit programma kleiner te maken doen we het volgende:

```
P=#2900

{ JSR#FFE6;STA#322;RTS

  0 2900 20 E6 FF JSR#FFE6
  0 2903 8D 22 03 STA#322
  0 2906 60       RTS
```

Dit programma neemt slechts 7 byte's in beslag. Het nadeel is echter dat men het programma kan overschrijven met BASIC. Om dit te voorkomen kunnen we het "Sinclair" principe toepassen n!. het plaatsen van de machinetaal in een REM statement. Dit gebeurt weer aan de hand van een voorbeeld. De punten moeten worden voorgesteld als spatie's.

```
10 REM.......
20 END
```

Daarna vraagt men een hexdump op om de plaats te bepalen waar de machinetaal moet beginnen.

```
X.#2900,
                     R  E  M  .  .
2900: 0D 00 0A 52 45 4D 2E 2E

2908: 2E 2E 2E 2E 2E 0D 00 14
       E  N  D
2910: 45 4E 44 0D FF 00 00 00
```

Daarna wordt de P op de goede lokatie gebracht. In ons geval op #2906.

```
P=#2906

{JSR#FFE6;STA#322;RTS
0 2906 20 E6 FF JSR#FFE6
0 2909 8D 22 03 STA#322
0 290C 60       RTS
```

Als we het programma listen dan zullen er in het REM statement vreemde symbolen staan. Het programma uit de REM is nu aan te roepen door LINK#2906 .
Men moet er wel rekening mee houden dat er maximaal 64 tekens geplaatst kunnen worden. Ik hoop dat ik een heleboel mensen weer een beetje verder heb geholpen in de wereld van de atom, en mochten er nog vragen zijn dan zal ik op de bijeenkomst deze proberen te beantwoorden.

Fred Danckaarts

De kast v/d ATOM

De geheugenkaart en de schakelkaart kunnen aan een flatkabel buiten de kast worden gebracht; echter binnen de kast gaat ook. Alleen past de kast dan niet meer. Ik heb nu gekozen voor een geheel 'nieuwe' onderkast waarmee de estetische vormgeving en de doelmatigheid v/h apparaat behouden blijven of zelfs verbeteren.

e.e.a. gaat als volgt.
Stukken kunststof worden verwerkt volgens onderstaande maten.
De dikte moet 5mm zijn, liefs niet meer of minder.
Vervolgens wordt de diaprojector tijdelijk gesloopt en wordt een stuk verwarmings element van ca. 500mm (4 Ohm/m) tussen twee klemmen gespannen en aangesloten op de 12V. voeding. Ik ga uit van een 100-150W trafo.

Plaat een wordt op de gemarkeerde plaatsen onder het element geplaatst, ca. vijf minuten verhit en haaks omgezet. Dit uiteraard tweemaal. Na pas en meetwerk worden de stukken twee en drie met de aangeven verbindmiddel 'gelast'.
Nu moet met schuurmiddel de haakse hoeken nog worden weggewerkt, en de zaak is bekeken.

De bak is dan 37mm diep. Reken erop dat het pas na wat oefenen lukt, maar het is zeker te DOEN.
Voor mensen die aan zo'n bak willen komen bestaat bij voldoende belangstelling de mogelijkheid om volgens dit principepe een kleine serie te fabriceren.
Dit kan in blank pvc of in perspex.(doorzichtig kunststof).

Bestelling geschied door overmaking van ƒ40 voor pvc en ƒ47.50 voor prspex op giro rek.3457629 tnv. Monsanto F. te Zwolle onder vermelding 'ONDERKAST' en evt. tel.nr.
Sluitingsdatum voor deadline volgende Acorn nieuws. Levering ca. 14 dagen na ontvangst.

Fernando,
Biesbos 153,
8032 VD ZWOLLE.

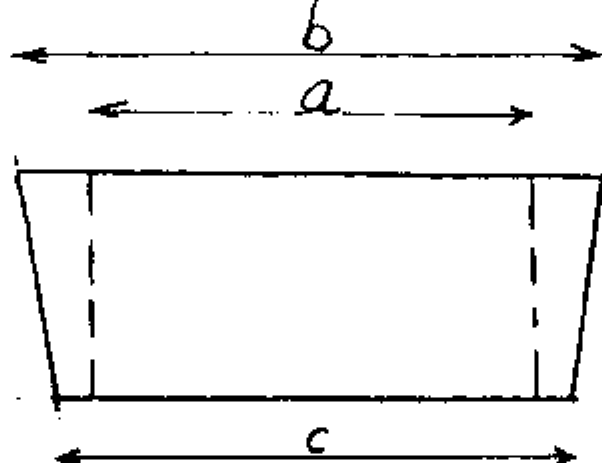
pvc lasmiddel: pvc lijm nr. 35 van Bison.
perspex lasmiddel: chloroform.
evt. nadere info.: Gerhard Visser.
zie ook de bouwtekening op pagina 42 !

++

GEINVENTEERD BEELD.

Voor een geinventeerd beeldscherm kun je op de plaats van link 4 (schema Atom) een inverter plaatsen. Met een schakelaar erbij kun je schakelen tussen normaal en geinverteerd beeld. Dit kan ook software-matig via een I/O-poort (b.v. VIA) en een Exclusief-Or-Poort.
Als de I/O-poort op 0 staat is het beeld normaal.

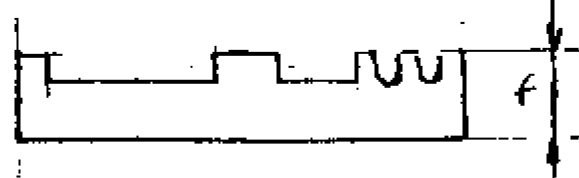
Uit "Acorntjesbrood" '83,1.



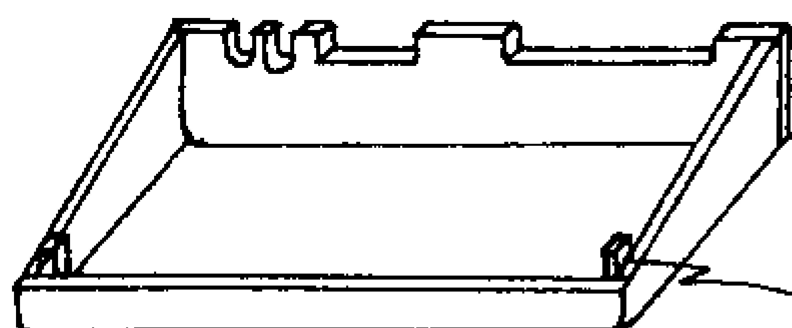
Plaat 1:
Bodem + korte zijkanten
(korte zijkanten langs
stippellijnen haaks omzetten)



Plaat 2: Voorzijde



Plaat 3: Achterzijde

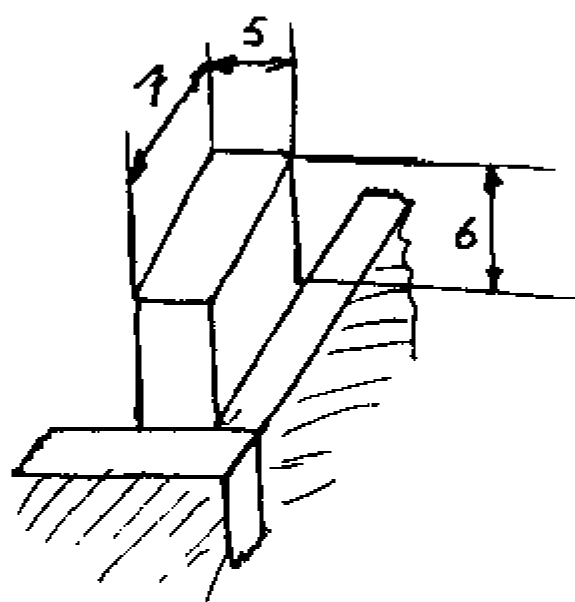


SAMENSTELLING

centrering(2x)

a = 376 mm
b = 524 mm
c = 446 mm
d = 230 mm
e = 42 mm
f = 75 mm

DETAIL
centrering



"De bakken zelf zien er gaaf uit, ze zijn voorzien van een goede centrering van de computer. Boven- en onderbak worden niet door schroefjes met elkaar verbonden. Hierdoor is zeer eenvoudig toegang mogelijk tot de AP-schakelaars etc."

Aldus de auteur.

SHAPESHAPER

Jan Wijnen

Programma om het construeren van een shape-tabel te vereenvoudigen.
Het programma maakt gebruik van een joystick om de tekening in elkaar te zetten.

```
10 X=128;Y=96;T=#3500;N=0;DIMN3;N?0=0;N?1=1;I=0
20 L=#B001;E=X;F=Y;J=1;?T=I
30 MOVEX,Y;S=T;M=1;GR.;?#E1=0
40 GOS.p;DOPLOT14,X,Y;WAIT;PLOT14,X,Y;KEYZ;WAIT;U.Z OR?L<253
50 IFZ=64 RUN
60 IFZ=69 G.250
70 IFZ=83 G.290
80 A=?L:255;B=A&30
90 IFA&64 M=M:1;DOU.?L=255;G.40
100 IFA&128 GOS.230;G.40
110 IFB<>2IFB<>4IFB<>8IFB<>16G.40
120 X=X+(B=8)-(B=2);Y=Y+(B=16)-(B=4);PLOT(12+N?(A&1)),X,Y
130 GOS.t;DOU.?L>253 OR M;G.40
140tREM vormen van de tabel
150 IFJ=1 I=I+1;?T=I;T?I=0;IF?T=0 GOS.n;R.
160 H=N?(A&1)*4;J=J:#11
170 IFB>2 H=H+3;IFB>4 H=H-2;IFB>8 H=H+1
180 T?I=T?I+J*H;R.
190n?T=255;P.$30"NIEUWE PAGINA"$7;T=T+256;I=0
200 LINK#FFE3;DOU.?L=255;X=E;Y=F;J=1;R.
220 REM controle
230 GR.;?#E1=0;MOVEE,F;G=S-256
240 DOG=G+256;SHAPEG;U.?G<255;MOVEX,Y;R.
250 REM einde
260 @=4;P.$30"SHAPE TABEL VAN:"&S'"                    TOT:"&(T+?T)'
270 P."AANTAL BLOKKEN :"(T-S)/256+1';E.
290 REM stap voor stap
300 T=S;N?2=1;I=0;J=1;GR.;MOVEE,F;?#E1=0
320 P.$30''"STAP VOOR STAP"
330 GOS.p;DOKEYZ;U.Z;IFZ=CH"T";?T=I;GOS.230;G.40
350 I=I+1;IFI=256 I=1;?T=255;T=T+256;MOVEE,F
360 N?3=T?I;SHAPE(N+2);X=?#5A;Y=?#5C;MOVEX,Y;G.330
370p@=4;P.$30"  X   Y  BYTES:"'X,Y,I;R.
```



Gebruiksaanwijzing voor Shapeshaper

Dit programma zet de tekening die je met de joystick tekent om in een tabel die door het SHAPE statement van de Josbox geinterpreteerd kan worden.
De plaats waar de tabel in het geheugen komt te staan wordt bepaald door de variabele T.

Voor het tekenen met de joystick zijn er de volgende voorzieningen aangebracht.

- @ : wis de tekening en begin opnieuw.
- E : einde,de tabel-lengte wordt precies aangegeven.
- CTRL : schakel tussen lijnen trekken en punten tekenen (wenkel/repeat)
- SHFT : controle van de tabel.
- S : ga naar Stap voor stap.

De knop op de joystick wordt gebruikt om te kiezen tussen twee plotmodes. Knop los geeft bij bewegen van de knuppel de eerste en knop in de tweede. Het instellen van de plotmodes gaat d.m.v. de variabele N.Hierbij geeft N?0 de plotmode aan met knop los en N?1 die met knop in.
Er zijn vier plotmodes om in te stellen: 0 - move 2 - invert
1 - set 3 - reset
Zie verder de gebruiksaanwijzing van het SHAPE commando.

Stap voor stap

Na het indrukken van de S ben je in de s.v.s. routine beland.Elke keer als je op de spatiebalk drukt wordt er een geheugenplaats uitgetekend en er dus twee puntjes op het scherm gezet.
Druk je dan op T dan kun je gewoon verder tekenen aan wat er tot dan op het scherm is verschenen.
De s.v.s. biedt de mogelijkheid om een tabel van band te laden,deze tot het einde uit te laten tekenen en er dan verder aan te werken.

Bewaren op band

Er zijn geen routines ingebouwd om de tabel op band te zetten.Dit moet de gebruiker zelf doen.Het gaat uitstekend als je na op E te hebben gedrukt de tabel wegzet met : *SAVE"naam" (beginadres) (eindadres+1)
Het kan natuurlijk ook in FCOS,alle adressen hex. opgeven.

Dit zijn zo'n beetje de mogelijkheden.Een beperking tot twee plotmodes lijkt belangrijk maar valt in de praktijk wel mee.
Het SHAPE statement is reuze handig,als je de tabel eenmaal hebt.
Daarom dit programma.
Als er mensen zijn die interresse hebben in de kaart van Nederland in tabelvorm dan kunnen zij die bij mij krijgen.

DE JOSBOX

Jos Horsmeier heeft de leden van de landelijke Acorn Computer Club de door hem ontwikkelde toolbox - inmiddels algemeen als JOSBOX bekend - vrij ter copiëring aangeboden. Van dat recht is ook in onze club ruim gebruik gemaakt. Er zullen nog maar weinig Atoms zijn, waarin de Josbox ontbreekt.
In "Acorn Nieuws" heeft Jos de instructieset van zijn geestesprodukt toegelicht. Daarin ontbreekt de betekenis van de foutmeldingen (error codes).

47. Fout in de lijst met argumenten bij PLAY.
97. Na RESTORE geen DATA aanwezig.
101. Interrupt (in de STEP-routine), die niet veroorzaakt wordt door timer 1 van de VIA (de enige, die is toegestaan).
154. HDUMP,XDUMP,DISAS,COPY,RELOC of STEP werden ingegeven anders dan in de direct mode.
193. Waar een adres moest worden opgegeven, is dat niet gedaan.
248. Na COPY of RELOC stond geen adres in het RAM-geheugen.
255. Verkeerd(e) argument(en) opgegeven bij RENUM.

De Josbox gebruikt van pagina 0 de adressen ≠ 90 tot ≠99. Aleen de laatste twee plaatsen zijn constant in gebruik als in een programma READ, DATA en RESTORE voorkomen. Deze adressen bevatten de file-pointer.

Testprogramma. Als het volgende programma'tje wordt ingetypt, dient na enige tijd OSEW op het scherm te verschijnen:

```
X=0 (return)
F.A=≠A000 TO ≠AFFF;X=X:?A;N.A (return)
P.X,$≠AF1C (return)
```

Joop Glasbergen

big benny

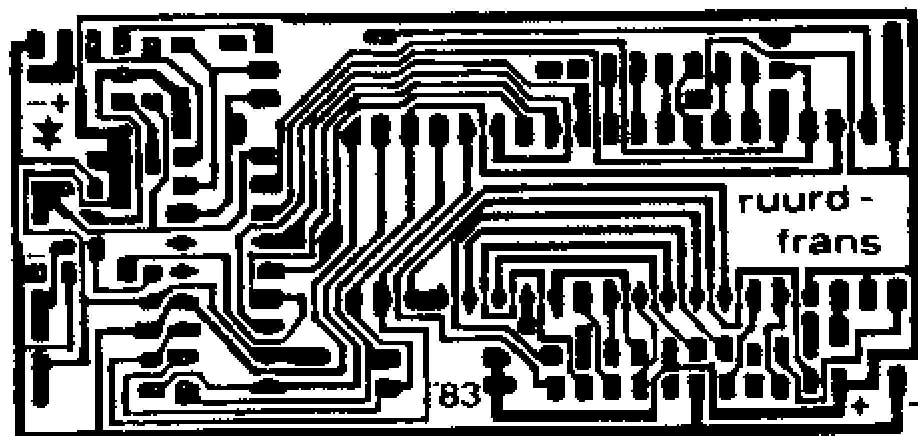
door: Ruurd van der Zee
en Frans van Hoesel

Wat zegt u?
Hoe laat 't is?
Of m'n computer dat ook weet?
Jazeker, met deze REAL-TIME CLOCK/CALENDAR weet hij zelfs de datum.

Hij wordt gevoed door twee - eventueel oplaadbare - batterijen en is voorzien van een eigen PIA. Er wordt dus geen aanspraak gemaakt op 'vrije' VIA-poorten. Van deze PIA zijn nog zes poorten over, waarvan er echter 2 (PB4 en PB5) gereserveerd zijn voor toekomstige uitbreidingen (alternatieve schakelkaart; komt er aan!). De overige 4 poorten zijn beschikbaar aan een goedkope 8-pens connector (50 cent, TELEC, Groningen) en kunnen voor allerlei doeleinden worden gebruikt. We spreken echter af dat we er alleen signalen binnen de computer mee zullen schakelen (niet extern dus).

Het printje is - door vele avonden puzzelen - geoptimaliseerd op minimale afmetingen (4 bij 8½ cm) en wordt aangesloten op PL8. Deze connector is oorspronkelijk bedoeld voor gebruik met het Econet systeem, maar dat wordt door de meesten onder u toch niet gebruikt.

Het gebruikte klok-IC (ongeveer f16,-) houdt behalve de tijd in uren, minuten en seconden ook nog de datum, maand, jaar en dag van de week bij. Bovendien is het in staat om softwarematig gestuurd interrupts te genereren met tussenpozen van één seconde, één minuut of één uur.



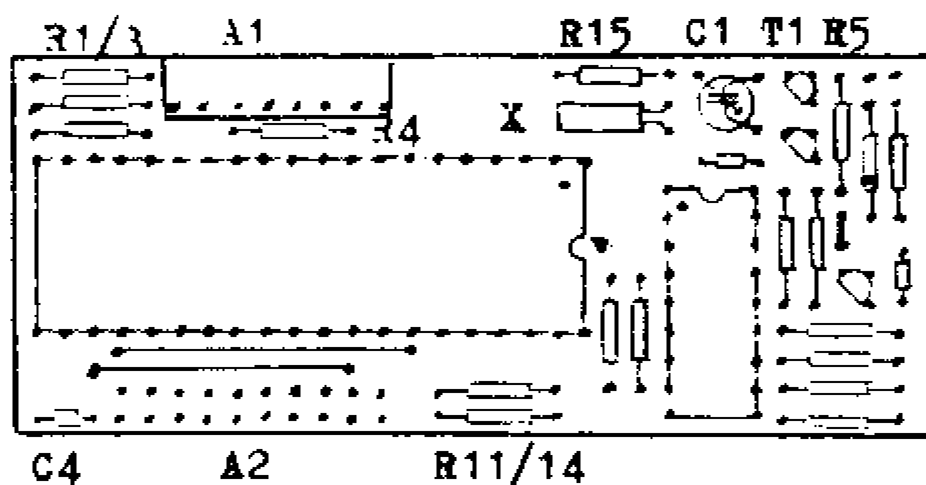
Het printje zal door de federatie worden uitgegeven. Het is dan van een soldeermasker voorzien, zodat ook minder ervaren clubleden het kunnen monteren. Bovendien komt er een klein pakketje met IC's (de MSM5832RS en de 6821) en misschien een connector voor PL8. De prijs van dit alles is nog niet precies bekent, maar ligt waarschijnlijk omstreeks de f35,-. De rest van de onderdelen (wat weerstanden, transistoren en een kristal) zijn in iedere electronica zaak verkrijgbaar.

Over het kristal nog iets bijzonders: als je een vriendelijke horlogemaker tegenkomt en je legt hem uit wat je aan het doen bent, dan is hij graag bereid je een oud (kapot) digitaal horloge te schenken, waar je dan voorzichtig het gratis kristal uit sloopt!

Voor de zelf-printers is elders in het blad een in spiegelbeeld afgedrukte printlayout op ware grootte (let's hope). De achterzijde van die pagina is blank, zodat je met transparant-spray kunt werken. (zie pag. 47)

Vóór het bouwen wordt van C1 eerst het kopje gehalveerd. Bovendien moet het solderen in de 3de t/m 7de rij van onder mooi vlak gebeuren. Bij het bouwen worden eerst de IC-voetjes gemonteerd, daarna de weerstanden en als laatste de transistoren. Let op de aansluitingen van de condensatoren en de diode. R16 MOET weggelaten worden als je geen oplaadbare batterijen gebruikt. De connector A2 wordt tijdens het solderen een beetje scheef gedrukt, zodat de bovenkant iets richting PIA neigt. De aansluiting voor de batterijen bevindt zich rechtsboven. Let op de drie draadbruggen.

Bij deze BIG BENNY hoort natuurlijk een programma dat de informatie van of de PIA omzet in iets bruikbaars. Het is een universeel bruikbare subroutine geworden. Als je iets wilt zetten moet de accumulator gelijk aan #40 zijn, om te lezen gelijk aan #80. Het X-register is #50 voor de tijd, #80 voor de datum en #60 voor de dag van de week. Om de tijd te zetten komt de informatie tijdelijk in de stringbuffer op #140. Vb: A=#40;X=#50;$#140="09:30";LINK #2800. Om de tijd weer te lezen dan: A=#80;X=#50;LINK #2800;P.$#140. Voor de datum: $#140="21-07-83" en voor de dag van de week: $140="5" (vrijdag).



D	:	AA119
R1/14	:	10k
R15	:	5k
R16	:	560 – 1k2
R17	:	51k
C1	:	trimmer 22p
C2	:	5p6
C3/4	:	tantaal 1u
T1/2	:	BC547 B
T3	:	BC557 B
IC1	:	6821
IC2	:	MSM5832RS
A1	:	haaks fem.
A2	:	female
X	:	32768 Hz

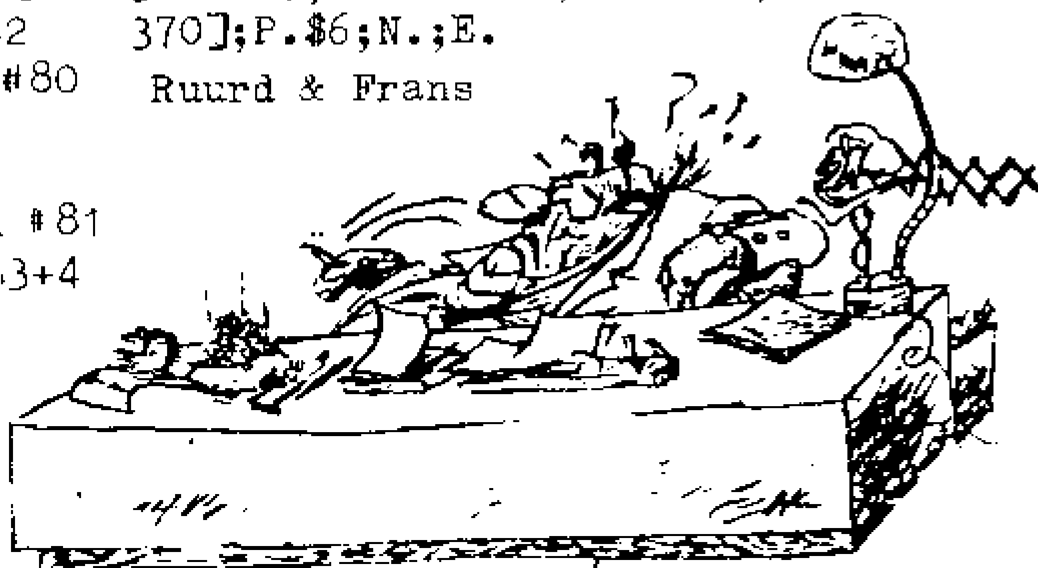
Voor gebruik op deze manier mogen de regels 290 t/m 360 worden weggelaten. Behalve deze standaard-subroutine hebben we ook maar alvast wat statements geschreven. Het zijn de statements TIME, DATE en DAY. TIME begint bij label LL4, DATE op LL4+3 en DAY nog weer drie plaatsen verder. In plaats van een syntaxdefinitie geven we een paar voorbeelden:

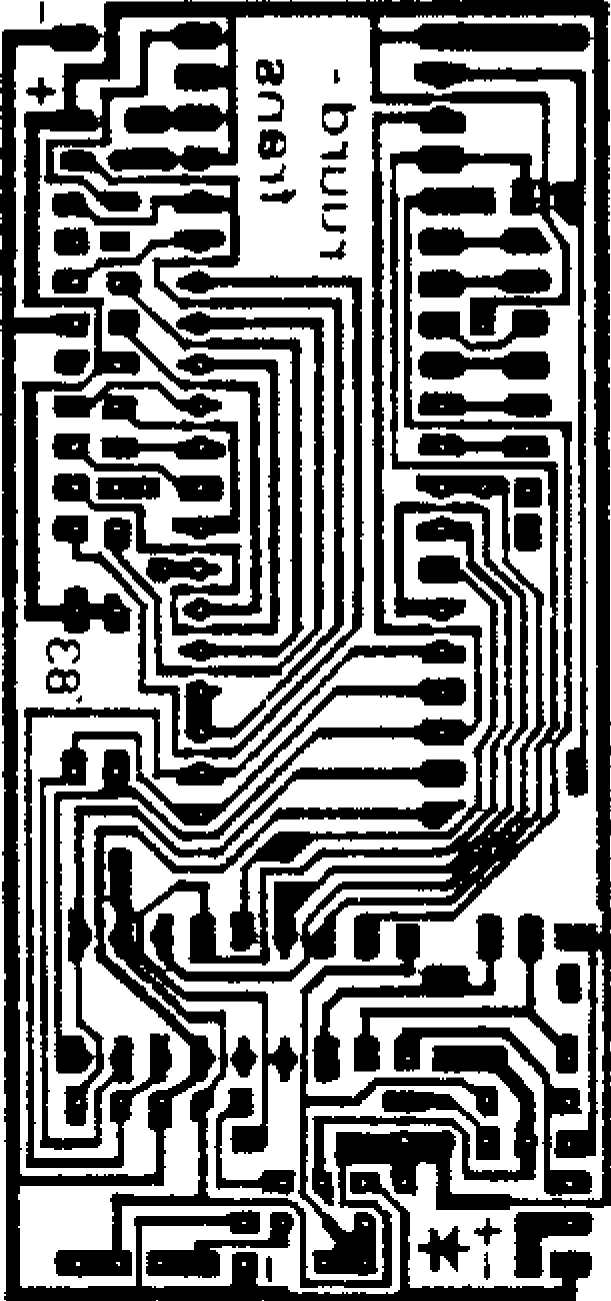
TIME="09:30" of TIME=$T worden gebruikt om het klokje op tijd te zetten.
TIME $A of TIME $#8200 haalt de tijd uit BENNY en zet die in een string.
TIME of TIME' drukt de tijd af; na het woordje TIME wordt verder gegaan alsof het een printstatement betrof (vandaar dat TIME' ook werkt). DATE en DAY gaan net zo en voor de laatste spreken we af dat DAY="0" overeenkomt met zondag. Regel 50 mag nu worden weggelaten.

Kunt u uw computer nog geen statements leren, heb dan nog even geduld. Er is nl. een EPROM in de maak, die op de plek komt van de josbox oid. en u in staat stelt om op zeer eenvoudige wijze, zonder omprogrammeren extra statements te maken, zonder schakelkaart.

```
10 REM BIG BENNY
20 DIM LL7;B=#B400;S=#52
30 FOR I=0 TO 7;LLI=999;N.
40 P.$21;F.I=1TO2;P=#2800;[
50 STA #80;STX #81
60:LL0;LDA @#40;STA #52;LDA @1
70STA #53;
80:LL1;LDA B+1;PHA;LDA B+3;PHA
90LDY @0;STY B+3;LDA B+2;ORA @#C0
100STA B+2;LDX @4;STX B+3;LDA B+2
110AND @#3F;STA B+2;PHA;LDA @#34
120STA B+3;STY B+1;LDA @#FF;BIT #80
130BPL P+4;LDA @#F0;STA B;STX B+1;
140:LL2;LDA (S),Y;AND @#0F;ORA #81
150LDX #81;CPX @#50;BNE P+4;ORA @8
160STA B;PLA;PHA;ORA #80;STA B+2
170LDA B;AND @#F;ORA @#30;CPX @#80
180BEQ P+6;CPX @#50;BNE P+4
190AND @#F3;STA (S),Y;PLA;PHA
200STA B+2;SEC;TXA;SBC @#10;STA #81
210INY;LDX @#3A;CMP @#50;BEQ LL3+4
220BCC P+6;ADC @#3F;LDX @#2D;CPY @2
230BEQ P+6;CPY@5;BNE LL3;STA #81
240TXA;STA (S),Y;INY;
250:LL3;CPY@8;BNE LL2;LDA @#FF
260STA B;PLA;ORA @#80;STA B+2;PLA
270ORA @#38;STA B+3;PLA;STA B+1
280LDA @#D;STA (S),Y;RTS;
290:LL4;LDX @#50;BIT #80A2
300BIT #60A2;STX #81;LDA @#80
310STA #80;JSR #F291;CMP @#3D
320BEQ LL7;CMP @#24;BNE LL6
330JSR #C78B;JSR #C3CB;
340:LL5;JSR #C4E4;JSR LL1;JMP #C55B;
350:LL6;DEC #3;JSR LL0;JMP #C39C;
360:LL7;JSR #CEB1;LSR #80;BNE LL5
370];P.$6;N.;E.
```

Ruurd & Frans





Inleiding.

De Schakelkaart van onze club is op dit moment in grote aantallen aanwezig bij onze leden en ongetwijfeld zijn velen op dit moment aan het experimenteren met een besturingsprogramma dat hun toolboxen, packs en eigengemaakte soft precies zó hanteert als zij dat graag zouden willen.
Het leuke van "onze" Schakelkaart is namelijk dat allerlei besturingsprogrammatuur kan worden toegepast, van zeer eenvoudig tot uitermate geraffineerd.
Zo kunnen alle in de handel als IC24 verkrijgbare EPROMS in ongewijzigde vorm worden toegepast. Je kunt ook eigenhandig ontwikkelde programma's in de Schakelkaart laden en met een zelfbedacht commando opstarten. Je kunt je schakelsoft zó maken dat alle statements van een nieuw ingeprikte EPROM onmiddellijk herkend worden; je kunt je schakelsoft ook zó maken dat bepaalde statements met "voorrang" behandeld worden, dus sneller. Je kunt....., je kunt nog veel meer, de mogelijkheden zijn vrijwel onbeperkt en worden alleen begrensd door de eigen fantasie.

Hiernaast is er nog de kwestie van het onderhoud van de schakelsoft. Je kunt natuurlijk op papier lijsten bijhouden van alle gerealiseerde statements; je kunt dit ook niet doen en alles trachten te onthouden. Vroeg of laat komt er dan een moment dat je twee statements hebt, die exact hetzelfde doen maar waarvan je de naam van de eerste vergeten was.........
De schakelsoft dient dus zèlf een lijst te kunnen produceren van alle actieve statements in de Schakelkaart. Bovendien zou het ook makkelijk zijn als de schakelsoft behulpzaam was bij het "aanleren" van nieuwe zelfgemaakte commando's.
En zo kunnen we nog wel een tijdje doorgaan.

In het hiernavolgende worden twee schakel-programma's beschreven door hun geestelijke vaders, nl. een SOS (=Schakelkaart Operating System) van Klaas de Raad en één van Bram Poot. Hopelijk vindt U hierbij iets van uw gading; wellicht vindt u hier juist dié zienswijze of aanpak uitgewerkt die u zelf ook nog 'ns wilde proberen.

Principieel.

Hiervóór werden reeds enkele mogelijkheden genoemd voor het hanteren van commando's en statements in de EPROMS op de Schakelkaart. Voor de duidelijkheid wil ik proberen hieronder een opsomming te geven van de verschillende mogelijkheden voor het hanteren van nieuwe commando's, packs en boxen, aanwezig op de Schakelkaart.

1. Successivelijk alle ROM-voeten aflopen, de één na de ander, totdat het gezochte commando is gevonden en kan worden uitgevoerd.
   Alle "Command-String-Interpretors" worden als het ware in serie geschakeld door het draai-schijf-programma (=de schakelsoft). Hierdoor wordt de computer traag, vooral als vaak gebruik wordt gemaakt van statements achteraan in de rij. Bovendien, wanneer bepaalde statements méér dan één keer in de CSI voorkomen, wordt altijd de voorste in de rij gepakt. Bijvoorbeeld DATA, READ en RESTORE komen vrijwel in elke toolbox voor. (Frans van Hoesel is daarom begonnen met het verzamelen van zoveel mogelijk handige statements in één EPROM, zodat er geen

doublures meer zijn. Bovendien kijkt deze EPROM ook naar eventuele statements in RAM; een mini-schakelkaartje dus. Zie "Ontwikkelingen" op pagina)

2. Maak je eigen Command-String-Interpreter waarin je alle te gebruiken statements opneemt en kies van de meermalen voorkomende statements de beste. Deze CSI vormt dus het hoofdbestanddeel van het draaischijfprogramma en is als het ware een selectie van het beste uit de aanwezige EPROMS.
   De schakelsoft is dan snel (de CSI is zo kort mogelijk) maar omvangrijk en bovendien worden de aparte CSI's in de EPROMS niet gebruikt. Zonde dus van die verloren EPROM-ruimte.
   Bovendien, de EPROMS moeten nu op een vaste plaats staan, zodat de afhandeling van een gevonden commando vanuit de draaischijf trefzeker kan worden ingeschakeld.

3. Zoals gebruikelijk, ligt de beste aanpak ergens tussen de bovenomschreven twee uitersten in.
   Je kunt bijv., als een bepaald programma toch alleen maar statements uit één toolbox gebruikt, alléén deze toolbox inschakelen, waardoor de CSI natuurlijk veel korter wordt.
   Je kunt de box met de meestgebruikte statements vóóraan zetten in de rij, zodat die het eerst aan de beurt komt. Of, als deze keuze te moeilijk is, haal dan alleen de meestgebruikte statements naar voren, door deze in een eigen CSI-tje op te nemen die je eerst doorloopt alvorens de toolboxen één voor één af te grazen.

De eerste methode had als voordeel, dat EPROMS geen vaste plaats in de Schakelkaart hoefden te hebben, want ze kwamen toch allemaal aan de beurt.

De tweede methode had dit voordeel niet: alle EPROMS moeten hierbij een eigen, vaste plaats hebben.

Bij de combinatiemethode 3 komen beide effecten voor: de EPROMS die vanuit het nieuwe CSI-tje worden aangesproken moeten een vaste plaats hebben, de andere niet. Een nieuw verworven EPROM kan dus zonder meer worden bijgeprikt, wat natuurlijk heel belangrijk is.

Opmerking.
Voordat ik de beide auteurs over hun SOS aan het woord laat, nog één opmerking.
Zoals u verderop zult zien gebruikt Klaas de Raad EPROMS die op enkele adressen zijn aangepast aan zijn schakelsoft, nl. op de eerste twee adressen moet staan # 40 en # E3 (i.p.v. resp. # 40 en # BF voor een "gewone" toolbox) en het terugspringadres aan het einde van de CSI moet zijn # E835 (i.p.v. # C558). Dit houdt dus in dat een "gewone" toolbox hier niet zomaar kan worden bijgeprikt en dat is erg jammer!
Klaas heeft hiervoor de volgende reden:

1. Dankzij de wijziging kan de schakelsoft nu onderscheiden welke box via de "eigen" CSI wordt ondersteund en welke boxen moeten "afgegraasd". Dit "grazen" gaat dus efficiënter, omdat de boxen met # BF vóóraan eenvoudig worden overgeslagen.
   Dit had natuurlijk ook gekund door de boxen, behorende bij de "eigen" CSI, in bepaalde voeten te zetten en het "grazen" tot de overige voeten te beperken, maar nu gaat het automatisch, dus altijd goed.

2. De in de club in zwang zijnde toolboxen worden reeds ondersteund uit de "eigen" CSI in het SOS. Toe te voegen EPROMS zijn dus waarschijnlijk eigen software-ontwikkelingen, waarbij de bovenstaande

aanpassingen eenvoudig worden "meegenomen" in het ontwerp. Bovendien, aldus Klaas, worden toolboxen meestal niet gekocht, maar binnen de club gecopiëerd. Ook dan is een kleine wijziging eenvoudig aangebracht.

3. Vaak moet een box sowieso worden aangepast om in de schakelkaart tekunnen werken. Meestal betreft het hier boxen die vectoren naar het A-blok richten. Wordt een andere EPROM in het A-blok geplaatst, dan gaat het natuurlijk mooi fout.

Voor al deze problemen zijn ook andere oplossingen mogelijk, dat bewijst Bram Poot's schakelsoft.

Afgezien van dit m.i. wat vergaand perfectionisme, heeft Klaas een SOS gebouwd met opmerkelijke mogelijkheden. De combinatie met de BOOTSTRAP (zie A.N. 1, '83) opent een reeks van nieuwe mogelijkheden en de inbouw van CHARON (zie Hobbit 1983, nr.4) van Frans van Hoesel, waardoor tien funtietoetsen mogelijk worden, verhoogt de toegevoegde waarde aanzienlijk.
Daarom wil ik u de SOS van Klaas niet onthouden.

Schakelkaart Operating System

Bram Poot

CHAIRPROGRAM is een minimumprogramma dat aangemerkt kan worden als de systeemsoftware van de schakelkaart, vergelijkbaar met b.v. de routine #FEFB zonder welke uitvoer naar de printer weliswaar mogelijk is, maar uiterst onhandig. Het programma neemt dus het omslachtige schakelen met de hand (?#BFFF=...) over.
Een beperkende voorwaarde die ik me bij de ontwikkeling ervan heb opgelegd was het Atom-principe waarmee toolkits worden opgenomen in het systeem. Ik vond dat hier niet aan geprutst mocht worden (niet iedereen kan dat namelijk).
Eerste eis: toolkits niet wijzigen.
Het ligt voor de hand om dan een hele grote tabel te maken met alle commando's en statements die voorhanden zijn, samen met epromnummer en executieadres. De nadelen zijn echter duidelijk: dubbel geheugengebruik en aanpassing noodzakelijk bij elke wijziging van de schakelkaartbezetting.
Tweede eis: geen tabel.
De draaischijf mag natuurlijk niet in conflict komen met andere programmatuur. Het verschijnsel dat de PPT uitgeschakeld moet worden als je de monitor wilt gebruiken is uit den boze.
Derde eis: Atom compatible (inclusief DOS).
Het aardige van een Atom is dat je op voorhand weet dat bedoelde systeemsoftware ook binnen deze eisen realiseerbaar moet zijn.

De Atom initieert na elke "return" een breakvector om eventueel optredende fouten (van allerlei aard) op te vangen. Eén van die fouten treedt op aan het eind van een toolkit als de interpreter niets meer heeft om het in behandeling zijnde statement mee te vergelijken. Wat de interpreter niet, maar wij wel weten is dat er naast die toolkit (d.w.z. op hetzelfde geheugengebied) nog een toolkit zit. De breakhandler moet dus in dat geval, en alleen in dat ene geval geen foutmelding ERROR 94 genereren, zoals gewoonlijk, maar de volgende toolkit inschakelen en de interpreter opnieuw een kans geven (JMP#A002).

Als alle toolkits behandeld zijn en het statement nog steeds onbekend is dan zetten we de breakhandler op origineel en komt alsnog de melding: ERROR 94.
Er zitten dus twee aspecten aan deze zaak:
1. het zetten van de breakvector
2. het successievelijk inschakelen van de toolkits
Punt 1. vindt zo snel mogelijk plaats na binnenkomst in de draaischijf (regel 160,170). T.b.v. punt 2. moet het byte #BFFF uitgelezen kunnen worden. Aangezien dit niet mogelijk is, wordt een kopie ervan gedefinieerd. Aan jou de keus waar en welk. In het kader van mijn verhaal in een vorige AcornNieuws over zero page gebruik mag ik in regel 60 natuurlijk niet "S=#B1" opschrijven. Verander dat dus even in "S=#41" of "S=#51" of zo. (Zie programma op volgende bladzijde.)
Op dit moment is de pure schakelsoft een feit en kunnen we gaan denken aan inbouw in een groter systeem. Eén faciliteit is al ingebouwd: als bit6 in het schakelbyte hoog is, is de draaischijf inactief (regel 150) en wordt er meteen doorgestuurd naar de eprom die door dit schakelbyte wordt aangegeven. Dit is gewoonlijk niet nodig, maar kan gewenst zijn waar snelheid belangrijk wordt. Het bitje wordt laag door shift return (regel 130), waarna de draaischijf weer driftig kan gaan schakelen.
Op de plek waar de breakvector zijn originele waarde terug krijgt kunnen nog tig andere dingen gebeuren. Bij mij gaat de interpreter hier via een achterdeurtje de PPT in. Vanwege de eerste en de tweede eis voldoet deze toolkit niet aan de derde. Daarom nam ik mijn toevlucht tot een lapmiddel, overigens tot volle tevredenheid zolang ik nog geen alternatief heb. Een andere mogelijkheid is: doorsturen naar de DOS, die de controle wel verder overneemt.
Voordat de breakhandler zijn nieuwe waarde krijgt (regel 160,170) kan eveneens van alles gebeuren. Bij mij wordt eerst een klein toolkitje met schakelstatements doorlopen (die per definitie niet op AXXX kunnen staan), in direct mode wordt de errorvector veranderd (naar een errorhandler met echte foutboodschappen), READ&WRITE CHAR-routines worden geïnitieerd met o.a. auto-repeat, keyclicks, verlengde inputbuffer, functietoetsen (anders dan CHARON; meer BBC-like d.w.z. wel of niet executeerbaar, ook midden in de regel te gebruiken), extra CTRL-karakters, vertraagd afdrukken, extra ster-commando's en een SET-UP-mode (voor de kenners: VT100-achtig). Deze zaken vallen echter qua principe niet onder het onderwerp schakelsoft en blijven hier derhalve buiten beschouwing.

Aangezien bij ieder goed verhaal een stroomschema behoort, heb ik er ook een gemaakt. Het bestaat grotendeels uit het bekende plaatje van de afhandeling van statements. Het gedoe met de breakvectoren is echter nieuw. Rechts boven zie je twee soortgelijke kolommen. De linker wordt doorlopen als de draaischijf actief is en kan in de ruit ERROR94 afgevangen worden. Als de draaischijf inactief is, **wordt** de rechter kolom doorlopen. (Stroomschema op pag.54)
Een opmerking: het getal I komt overeen met de inhoud van het schakelbyte S uit het programma.
Voor de rest zou dit schema voor zichzelf moeten spreken.

N.B. Bij de eerste opstart van de Atom moeten in het schakelbyte en in #BFFF een nul staan. Heb je een bootstrap dan kun je dat automatiseren, anders even met de hand doen.
Na elke volgende break van welke aard ook is geen actie nodig.

```
10 REM CHAIRPROGRAM
20 REM AUTEUR: BRAM FOOT
30 Q=#E000;REM BASISADRES
40 !Q=0;REM I.V.M. FOUTEN
50 ?#23=0;?#24=#B2;REM DIM POINTER
60 S=#B1;REM SCHAKELBYTE
70 E=#E400;REM UITBREIDINGEN NA AXXX
80 DIM BB9;P.$21
90 F.I=0 TO 9;BBI=Q;N.I
100 F.I=1 TO 2;P=Q+2
110[\DRAAISCHIJF
120 BIT#B001;BMIBB3 NORMAAL
130 LDA@0;STA S SHIFT INGEDRUKT
140:BB3 LDA S
150 BIT S;BVSBB2 OP SLOT
160 LDA@BB0&#FF;STA#202 BREAK-
170 LDA@BB0/256;STA#203 HANDLER
180 LDA@#FF;STA S 'ROM#-1'
190:BB1 LDX@#FF;TXS MAAK STACK SCHOON
200 INC S;LDA S
210 CMP@8;BCCBB2 VOLGENDE ROM
220 LDA@#D8;STA#202 ORG.BREAK-
230 LDA@#C9;STA#203 HANDLER
240\LDA E\CMP@#40\BNEBB6
250\LDA E+1\CMP@#BF\BNEBB6
260 JMP E+2 EXTRA UITBREIDINGEN
270:BB0 PLA;PLA;STA#0
280 CMP@#5E;BEQBB1 ERROR 94
290 JMP#C9DC NORMALE BREAKHANDLER
300:BB2 STA#BFFF
310\LDA#A000\CMP@#40\BNEBB6
320 LDA#A001;CMP@#BF;BNEBB6
330:BB8 JMP#A002 PROBEER MAAR
340:BB6 JMP#C558
350];P.$6;N.I
360 ?D=#40
370 END
```



Toelichting:

60: Het schakelbyte zou ≠ BFFF moeten zijn,maar dat is WRITE-ONLY omdat de ontwerpers dit soort programmatuur niet hadden voorzien en omdat de kaart anders nodeloos ingewikkeld zou worden. Je kunt nog 1K stapelen (≠BC00-≠BFFF) of een ander adres nemen b.v.
≠FE als je geen printer hebt of
≠BF als je geen DOS hebt of
≠3FD als je geen kleurenkaart hebt of
≠67FF op de geheugenkaart of
≠E001 op de schakelkaart
(zonder write protect voor de laatste twee mogelijkheden)

70: Kan ook ≠7000 zijn of ≠6800 of ≠E800 (of ≠0080, waar dan een makkelijk te wijzigen JMP-instructie staat of een JMP(indirect)) Kies maar uit.

240: Deze regels zijn niet echt nodig.

250: Als je echter niet weet of er extra uitbreidingen zijn, kun je ze beter wel meenemen.

310: De FP ROM test ook alleen maar op ?≠A001=≠BF

EXECUTEER STATEMENT

STATEMENT

BASIC OF FP?
JA
NEE

?#BFFF+I

BRAAI-SCHUIF?
JA
NEE

?#A001 =#BF?
JA
NEE

BRK

JMP #E002

OP SLOT?
JA
NEE

ZET BRKVEC
I ← -1

I ← I+1

I ≥ 8?
JA
NEE

?#BFFF+I

?#A001 =#BF?
JA
NEE

BRK

JMP (BRKVEC)

JMP #A002

OK?
JA
NEE

BRK

JMP (BRKVEC)

ERROR 94?
JA
NEE

JMP #A002

OK?
JA
NEE

BRK

JMP (BRKVEC)

ERROR

ZET BRKVEC

JMP VERDERE UITBREIDINGEN

OK?
JA
NEE

bij de beesten af



SCHAKELKAART OPERATING SYSTEM (SOS)

Het schakel operating system SOS, is een systeem, dat er voor zorgt, dat de verschillende handelsboxen * geplaatst op de schakelkaart, zonder problemen en zonder om te programmeren! met elkaar samenwerken.
Statements uit de verschillende boxen kunnen door elkaar gebruikt worden in één programma. Het SOS zorgt ervoor, dat automatisch de juiste BOX ingeschakeld wordt. Zie stroomdiagram op pag. 57 .
Het SOS bevat bovendien nog een aantal bijzonder gebruiksvriendelijke faciliteiten:

-Wanneer m.b.v. FCOS 1200 Baud wordt ingeschakeld, dan worden files vanuit b.v. ATOMCALC en WORDPACK ook netjes op 1200 BAUD naar cassette geschreven.

-Ook na het commando GRMOD, kunnen rustig statements uit andere boxen gebruikt worden.

-Unnamed files worden voorzien van een Check-sum byte.

-Functietoetsen zijn meteen beschikbaar, want het programma CHARON van F. van Hoesel is ingebouwd. Zie het meinummer van Hobbit.

-Ook het statement AUTO is ingebouwd en het piept netjes, wanneer een regelnummer al aanwezig is.

-Wil je een listing stoppen? Dat kan, met CTRL-S. Door CTRL-Q is de uitlezing weer te hervatten. (veel handiger dan de pagemode)

-Wil je zelf statements erbij maken: daar is in voorzien. Het is zelfs mogelijk om statements op de 16K kaart te plaatsen.

Het geheel neemt slechts 2K geheugen in beslag en wordt geplaatst in het gebied E800-EFFF. De ervaring leert, dat plaatsing in EPROM (2716) aan te bevelen is. Wanneer het in RAM geplaatst wordt, moet dit gebied zeer zorgvuldig beveiligd worden m.b.v. Wright protect!!!

Het systeem is te 'linken' met LINK≠E805. 1200Baud, Charon en (indien aanwezig) de Softtool worden nu ingeschakeld.

Automatisch inschakelen wordt mogelijk, wanneer de Bootstrapschakeling is ingebouwd. Dit geeft bovendien de volgende extra faciliteiten:

-BREAK samen met REPT 'linked' automatisch het systeem. De mogelijkheid is hierbij aanwezig om een gebruikersroutine aan te brengen, om b.v. automatisch een programma te runnen of in te laden etc. etc.

-BREAK samen met SHIFT veroorzaakt de normale 'Atom' BREAK.

-Alleen BREAK geeft slechts de mededeling BREAK, of tijdens een BASIC-programma BREAK AT LINE x, waarbij x het lijnnummer aangeeft, waar het BASIC-programma werd onderbroken. Belangrijk hierbij is, dat alle systeemvectoren en de pagepointer etc. blijven zoals ze waren. Ook een eventuele GRMOD blijft. Dus blijft het systeem 'vastzitten', b.v. in een cassetteroutine of in een machinetaalroutine, druk dan eenvoudig op BREAK!

In het hierna volgende gedeelte zullen we achtereenvolgens beschrijven:
-aanpassen F.P. ROM -zelf boxen maken -beschrijving RCK-BOX
-beschrijving van de statements, die opgenomen zijn in de SOS
-inbouw van de bootstrap -overzicht van gebruikte adressen c.q. geheugens.

De Floating Point ROM: Om de statements te herkennen moet uiteraard de z.g. 3-weg wissel worden aangebracht in de F.P. ROM. Gekozen is voor de volgende routine: adres in F.P. D4AF:

```
D4AF: LDA ≠E7FD
      CMP ≠E3     testbyte voor SOS
      BEQ ≠D4C0   op D4C0 staat JMP(≠E7FE) E7FE: →E800
      LDA ≠A001   etc.
```

* De bedoelde handelsboxen zijn: Wordpack,Atomcalc,Toolbug,Softtool en Josbox. Voor andere boxen is een aanpassing noodzakelijk.

Aanpassing van de F.P. ROM:

4BC	was 00	wordt F0		D4BC	was 03	wordt 83
4B1	A0	E7		D4C0	4C	6C
D4B3	40	E3		D4C1	02	FE
D4B4	D0	F0		D4C2	A0	E7

Voor de handelsboxen en ook de zelfgemaakte boxen gelden geen vaste plaatsen, uitgezonderd de JOSBOX, dit vanwege de snelheid. Ook een eventuele Softtool heeft een vaste plaats, om de *commando's niet te vertragen. Op deze plaats kan uiteraard rust[illegible] een andere box geplaatst worden.
-JOSBOX plaatsen op A-blok No.4
-Eventuele SOFTTOOL plaatsen op A-blok NO.3

Zelf te maken boxen moeten starten met 40 en E3 (testbytes) op repectievelijk A000 en A001 [illegible]r word automatisch gesprongen naar A002. Wordt een statement in deze box [illegible]t gevonden (einde tabel) dan niet naar C558 springen maar terug in het SOS [illegible] E835, zodat het systeem in andere boxen verder kan gaan zoeken. Een box va[illegible]it type is door mij gemaakt en bevat de volgende statements:

-CURSOR, HEX, ZERO, POP, BEEP, STOP, VAR en DELETE. Deze statements zijn bekend van de TOOLBOX van P.P. en zijn herschreven. De statements gebruiken geen 'vrije zero[illegible]ge' en testen netjes op einde statement(C55B). HEX is ingekort en is nu ook te gebruiken in combinatie met een printer of in GRMOD. VAR is sterk ingekort [illegible]n VAR≠ werkt nu wel correct.

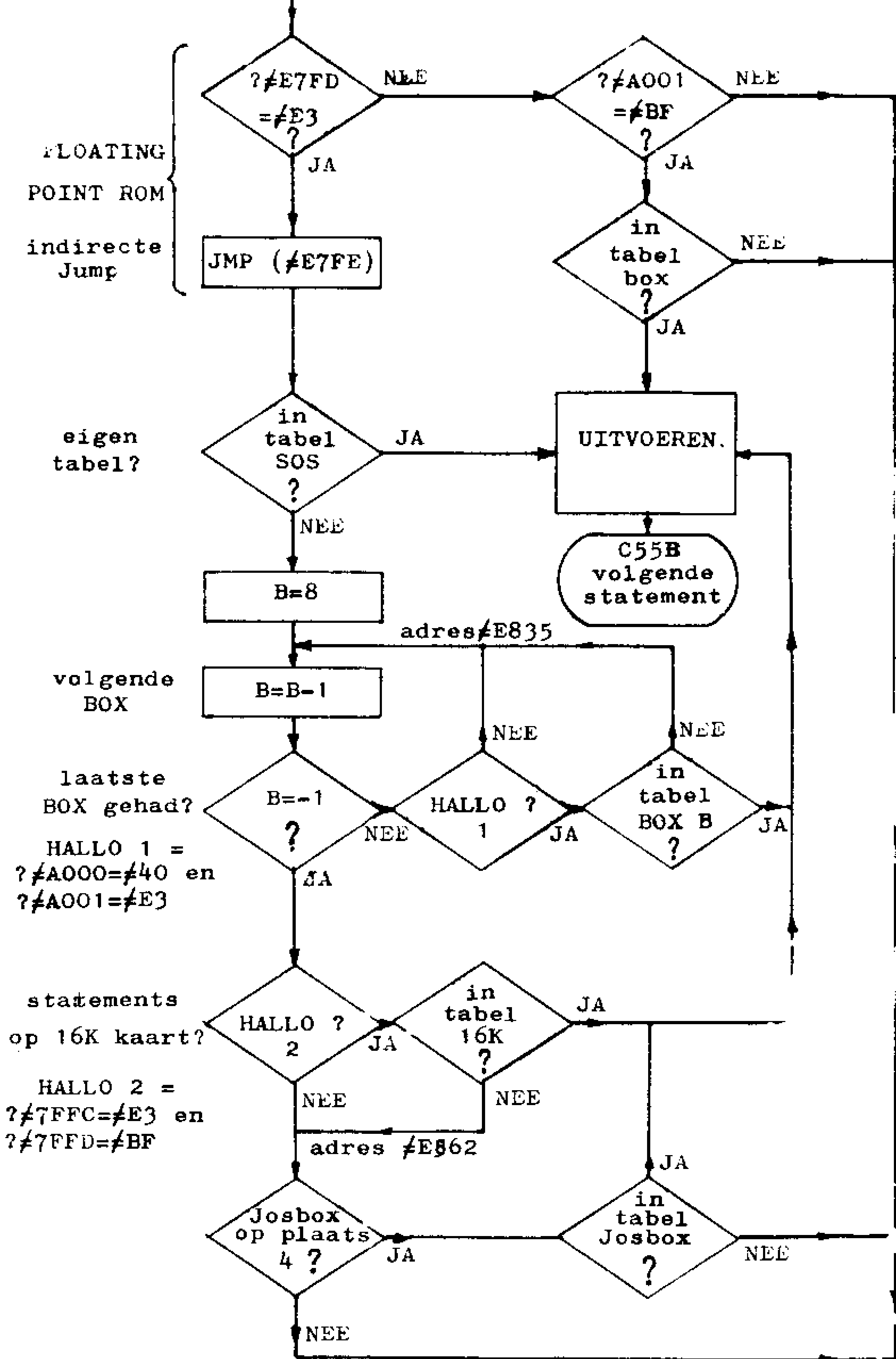
-ook opgenomen zijn de z.g. monitorcommando's (zie A.N. 7 1982 blz.38) HOME, HIGH en VHIGH stellen de tekstspacepointer in op resp. ≠29, ≠82 en ≠98. TP laat TOP zien en TS laat de tekstspacepointer zien.

-nieuw zijn de volgende statements BELL, INKEY wacht op toets , PAU[illegible] PAUSE 255 wacht 255/60 seconde (max). HTAB b.v. [illegible] 45 geeft 45 [illegible] op de printer), VTAB b.v. VTAB3 geeft [illegible] nieuwe regels (CR+LF). DPO en FPOP, voortijdig springen uit resp. DO..UNTIL en FOR..NEXT lus
VDU-B, bekend als Softvdu: grote letters en kleine letters in MODE 4. Gebruikt DATA uit de WORDPACK ROM, die verhuist wordt naar gestapeld geheugen 9C00-9FFF ! De routine zelf wordt verhuist naar E300-E4FF (z.g. kladblok)
VDU-W idem maar nu zwarte letters op een witte achtergrond.

-tenslotte MONITOR. Het betreft hier de bekende monitor van R.Heuvel. Verhuist zichzelf naar E100-E4FF. (kladblok) Deze lokatie is gekozen, zodat in alle boxen gekeken kan worden.

Beschrijving van statements, die in het SOS zijn opgenomen. De tabel omvat: FCOS, SCOS, CODE, AUTO, CALC, CALCR, EDIT, TEXT, EDT, DEBUG, CHARON, CHOFF, GRMODE, TXMOD en BOX.
-CALC en CALCR voor de CALC ROM
-EDIT en TEXT voor de WORDPACK ROM
-DEBUG en EDT voor de TOOLBUG ROM (gekozen is voor EDT ter onderscheiding van EDIT uit WORDPACK!)
De genoemde statements zoeken eerst of de betreffende box aanwezig is en springen daar dan in, anders volgt ERROR 94.
De statements FCOS, SCOS, GRMOD en TXMOD zijn opgenomen om de juiste vectoren aan te passen maar let op, ze werken alleen samen met de JOSBOX!

Verder:
-CHARON schakeld Charon in met standaard functieset op ≠E500. En b.v. CHARON≠87 geeft functieset op ≠8700.
-CHOFF zet Charon uit.
-AUTO werkt ongeveer als AUTO uit TOOLBOX P.P. echter met enkele verbeteringen. b.v. AUTO 50 nummert 50,60,70, ... AUTO ,50: 10,60,110, ...
I.p.v. 1 testbyte worden nu 2 testbytes gebruikt, zodat spontane regelnummering nu niet meer voorkomt. Escape schakelt nummering nu wel uit.
Let op, omdat AUTO geïntegreerd is in CHARON, wordt CHARON automatisch ingeschakeld door AUTO indien CHARON uit was.
-BOX b.v. BOX 4 schakelt box 4 (JOSBOX) in. Bij een getal groter dan 127 volgt ERROR 227.



{≠E7FD,≠E7FE en ≠E7FF worden bij
REPT BREAK AUTOMATISCH INGESTELD
OP resp. ≠E3,≠00,≠E8

(C 58
Erro 94
of ";" of ≠OD)

-CODE dit is een "FIND"-instructie voor machinecode.
Aan de hand van een voorbeeld zal ik de werking van dit statement verklaren.
BOX 6; CODE ≠A000,≠B000
We hebben hiermee BOX 6 ingeschakeld en gaan zoeken tussen het opgegeven start en eindadres. De computer zal antwoorden met: "INSTRUCTION?"
We kunnen nu 1 of 2 losse bytes opgeven b.v. ≠FFE3 of 1 of meerdere assembler instructie's, die vertaald worden. b.v. LDA @6; JSR ≠FFE3. In beide gevallen zoekt de comuter of deze combinatie ook voor komt in het opgegeven gebied, en geeft het adres of de adressen, daarna volgt weer "INSTRUCTION?"
Worden er teveel assemblerinstructies opgegeven (te lange regel) dan volgt ERROR 3. (bij zoeken naar 2 losse byte's wordt gezocht naar lowbyte resp. highbyte dus in bovenstaand voorbeeld E3FF)

Het inbouwen van de 'Bootstrap'.

Alereerst voor de duidelijkheid een uitleg, wat de functie is van de bootstrap beschreven in A.N. febr. '83.
De 6502 processor kijkt, nadat hij gereset wordt (BREAK) naar FFFC(lowbyte) en FFFD(highbyte) , springt naar dit adres en gaat met uitvoering beginnen.
We willen graag dit adres veranderen, maar helaas staan deze bytes in ROM.
Met de bootstrapschakeling houden we de processor voor de gek, door wanneer de processor wil lezen of schrijven naar de 8 hoogste bytes (FFF8 t/m FFFF) één adreslijn laag te trekken, in het oorspronkelijke ontwerp A15. Deze 8 bytes komen nu overeen met 7FF8 t/m 7FFF. Deze adressen staan in RAM en kunnen dus wel veranderd worden. Opgelet: onze Acorn atom heeft helaas de neiging (waarschijnlijk door looptijdvertragingen) bij het lezen van deze hoge adressen ook te schrijven, waardoor de inhoud verandert. Dus altijd Wright Protect!!
Volgens mijn filosofie is het echter niet zo netjes, om de 16k-kaart te misbruiken voor dergelijke doeleinden, wanneer we een schakelkaart aan boord hebben, met ruimte op het E-blok.
We kunnen i.p.v. A15 ook A12 laagtrekken. FFF8 t/m FFFF komen nu overeen met EFF8 t/m EFFF.
Het schema van de Bootstrap is getekend in fig.3. We zien een 13input NAND. Hierop zijn 12 adreslijnen om te testen, of er inderdaad naar de hoogste 8 bytes wordt gekeken. Adreslijn 12 hebben we onderbroken met een AND, waarmee we onder bovenstaande voorwaarde A12 laag kunnen trekken (A12')
Om de adressen uit het E-blok te krijgen, moeten we het bootstrapprintje iets veranderen.

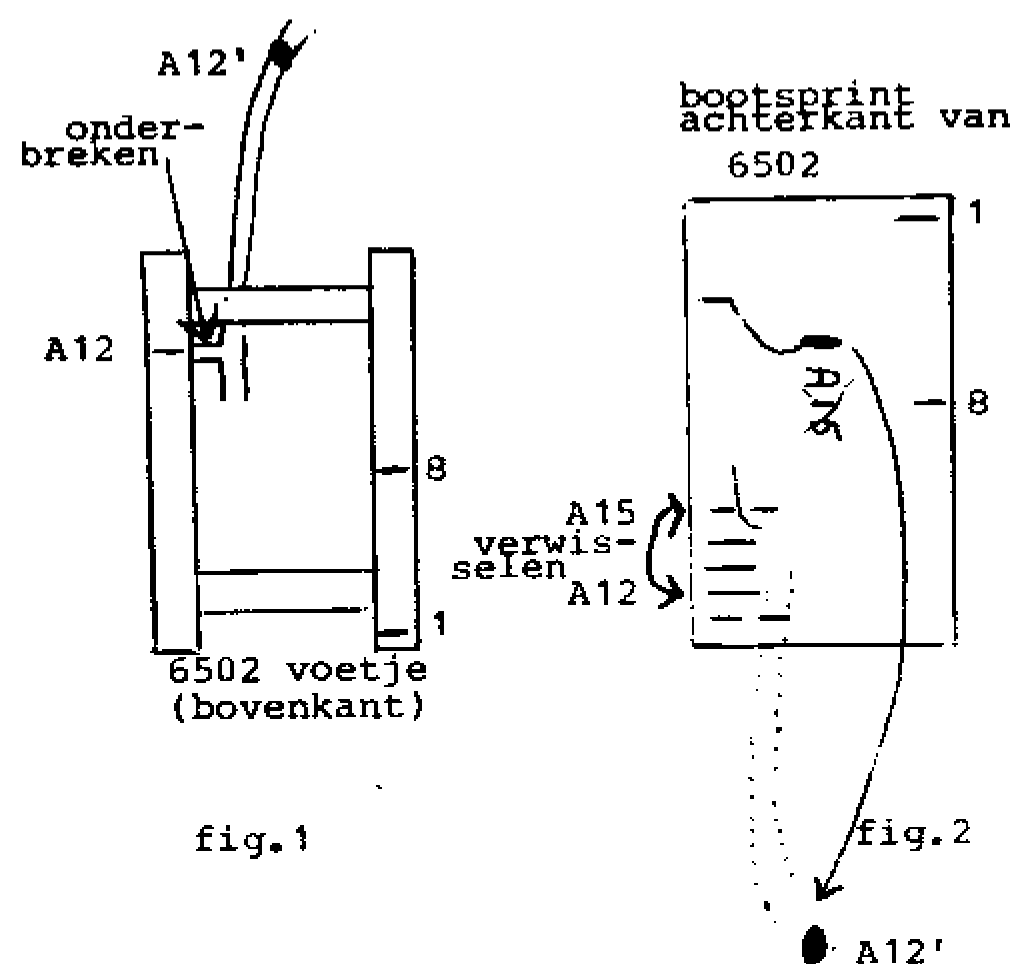


fig.1
fig.2

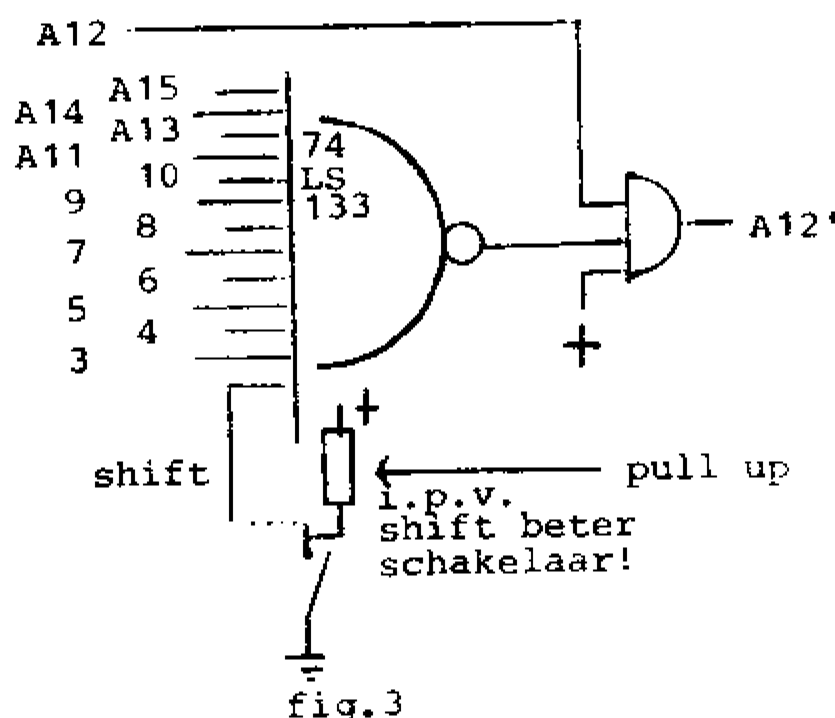


fig.3

-Verwissel de 2 adresdraden A12 en A15, die het bootstrap printje met de 6502 verbinden. (zie fig. 2)
-Maak nu géén onderbrekingen in adreslijn A15! , maar nu in adreslijn A12. We kunnen nu met één onderbreking volstaan. We halen de processor uit zijn voetje en zoeken punt A12 op. (zie fig.1) We onderbreken het printspoor bij het aangegeven pijltje. (doorkras en) We noemen het losgekoppelde printspoor A12'. De uitgang van het printje verbinden we hiermee, (was A15 op printje) (zie fig. 2). Deze verbinding maken we natuurlijk aan de achterzijde.
-Opmerking over het punt shift (zie fig.3):
Het punt shift is er voor bedoeld, om de bootstrap buiten werking te zetten, waardoor bij indrukken shift de acorn zich weer normaal gedraagt. Handiger is in dit geval, om een aparte schakelaar te plaatsen (zie fig.3).

Gebruikte adressen c.q. geheugenplaatsen: (alle adressen hexadecimaal)

De geheugenplaatsen zijn met zorg gekozen, om conflicten tussen de verschillende boxen te voorkomen en om zo zuinig mogelijk vrije geheugenplaatsen in beslag te nemen.

23D,23E en 23F zijn gereserveerd voor gebruik in de Softtool.	
239,23A,23B en 23C	worden gebruikt in CHARON als geheugenplaatsen voor OSWRCH en OSRCH. (In de originele versie van CHARON zijn dit andere adressen)
238	is highbyte functieset. Normaal E5
237	opslag BFFF (om te kunnen lezen)
236	tijdelijke opslag 237 i.v.m. *commando's softtool.
235	tijdelijke opslag 237 i.v.m. GRMOD.
233 en 234	testbyte's t.b.v. AUTO.
AA t/m AF	worden door de softtool gebruikt.
AA t/m A0	worden ook gebruikt in het statement AUTO.

RCK-BOX: Alleen de statements VDU-B en VDU-W gebruiken vrije zeropage n.l. 9B t/m 9F!
De monitor (R.H.) gebruikt uiteraard ook zeropage n.l. 90 t/m A9

7FFC en 7FFD	testbytes voor eventuele statements 16k-kaart. (E3 en BF)
7FFE en 7FFF	het adres van de 16k-kaart statement interpretor.
E7F9 en E7FA	testbytes voor een aventuele boots-gebruikersroutine. (E3 en BF)
E7FB en E7FC	het adres van de bootsgebruikersroutine.
E7FD	testbyte voor de F.POINT ROM. Wordt bij opstarten SOS automatisch E3 gemaakt.
E7FE en E7FF	adres voor de indirecte JMP van de F.P. Wordt bij opstarten SOS automatisch E800 gemaakt.
E000 tot E0FF	vrij voor een eventuele DOS routine (doorschakelen 2e Eblok)
E100 tot E4FF	'kladblok' voor MONITOR, VDU-B en VDU-W. (RCK-BOX). Deze ruimte kan dus ook tijdelijk voor andere doeleinden gebruikt worden!
E500 tot E77F	hier bevindt zich de standaard functie-set voor CHARON.
E780 tot E7F8	niet in gebruik.

Tenslotte: De RCK-BOX heeft nog aardig wat ruimte over. Ter herkenning zijn de lege plaatsen opgevuld met ≠77. Deze ruimte kan nog met nieuwe statements worden opgevuld, zoals een TRACE routine voor BASIC of WHILE/WHILEEND etc. Wie helpt er mee ontwikkelen?

Klaas de Raad

ontwikkeling

Momenteel zijn in ontwikkeling twee nieuwe hulpmiddielen voor de ATOM die in principe bedoeld zijn voor die mensen die nog niet over een schakelkaart beschikken. Ze zijn geheel onafhankelijk van elkaar te gebruiken en samen maken ze een dure schakelkaart vrijwel geheel overbodig. Wel wordt u verondersteld over een 16k-kaart te beschikken. Het gaat over de volgende twee vindingen:

1) Een nieuwe EPROM op Axxx. Nu zult u zeggen alweer een nieuwe? Ja, maar deze beschikt over een aantal bijzondere eigenschappen die hem uniek maken. Uiteraard zitten er de meest gebruikte statements in zoals READ, DATA, RESTORE, KEY etc. maar ook enkele nieuwe. Zo zit er een zeer krachtige editor in, die het verbeteren van programma's veel gemakkelijker maakt (ook handig voor het 'aanpassen' van hobbyscoop-programma's) en zit er ook een heus procedure statement in. Wat er niet in zit zijn de meer gespecialiseerde statements zoals SHAPE, BLOCK, PLAY (wel BEEP) ed. Dit is echter niet een gemis want let op: deze EPROM is zodanig georganiseerd dat nieuwe statements zeer gemakkelijk in RAM kunnen worden geladen. Heeft u dus op een gegeven ogenblik een stel grafische statements nodig, dan pakt u het bandje met grafische statements en laadt dat graphics-pack gewoon even in RAM en klaar is kees. Ook het zelf schrijven van statements is nu eenvoudiger geworden, omdat u niet een nieuwe interpretator hoeft te schrijven. Slechts een lijst met statements en hun adressen is voldoende om de ATOM die statements te 'leren' (meerdere lijsten kunnen ook worden gebruikt). Als toetje: voor al die extra statemnts hoeft u niet eens de FP-ROM te wijzigen.

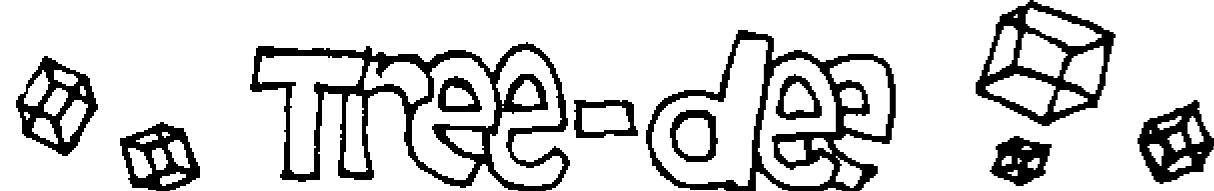
2) Een miniatuur schakelkaartje dat in ieder geval kan schakelen tussen één EPROM en 4k CMOS op Axxx. De gedachte hierachter is dat het voor de meesten onder u voldoende is dat indien nodig ATOM CALC of WORD-PACK ed. in RAM kan worden geladen. Deze RAM is uiteraard voorzien van battery-backup. Bovendien zullen vele onder u aan één EPROM met extra statements ruim voldoende hebben. Kaartje past uiteraard in de kast.

Samen zijn ze nog krachtiger doordat nu vrijwel alles dat u met de schakelkaart had willen doen mogelijk is geworden (en dat voor veel minder geld). Wilt u ATOM CALC gebruiken, dan kan dat; extra statements, nou ook dat is mogelijk. Heeft u echter haast omdat u direct wilt kunnen switchen tussen bv. CALC en WORD-PACK of omdat u geen tijd heeft om deze ontwikkeling af te wachten, dan moet ik u toch aanraden om de schakelkaart te kopen.

De ontwikkeltijd is tamelijk lang, doordat alle stements voor de nieuwe EPROM opnieuw gescreven moeten worden, opdat ze zeer kort worden (dan passen er veel meer in één EPROM). Het miniatuur schakelkaartje wordt in samenwerking met Ruurd van der Zee ontworpen.

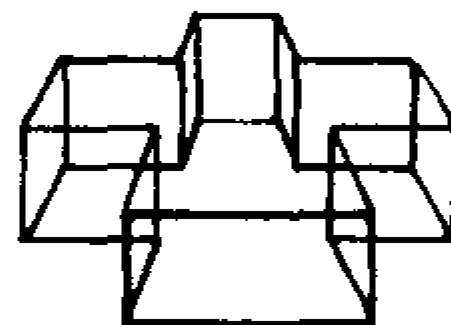
Groetjes, Frans.






```
   0 REM 3D TEKENEN
  10 P.$12"     ** Threedee **"'
  20 FDIM %XX2,%YY2,%ZZ2
  30 FIN."Azimuth "%A;%B=%A
  40 FIN."Elevatie"%E;%F=%E
  50 FIN."Afstand "%D
  60 FIN."Azimuth-verandering "%G
  70 FIN."Elevatie-verandering"%I
  80 IN."Vergrotingsfactor"F
 100b CLEAR4;T=0
 110 %A=%A*2*PI/360
 120 %E=%E*2*PI/360
 130 %ZZ2=SIN%E;%XX2=COS%E*SIN%A
 140 %YY2=COS%E*COS%A;%XX0=-COS%A
 150 %YY0=SIN%A;%ZZ1=COS%E
 160 %XX1=-SIN%E*SIN%A
 170 %YY1=-SIN%E*COS%A
 180 RESTORE
 200d READX,READY,READZ
 210 %W=%D/(%D-%XX2*X-%YY2*Y-%ZZ2*Z)
 220 %V=%W*(%XX0*X+%YY0*Y)
 230 GOS.a;%H=%V
 240 %V=%W*(%XX1*X+%YY1*Y+%ZZ1*Z)
 250 GOS.a
 260 IF T>0 G.c
 270 PLOT13,(%(F*%H)+128),(%(F*%V)+96)
 300c IF T>0 DRAW(%(F*%H)+128),(%(F*%V)+96)
 310T=T+1; IF T<42 G.d
 320 %B=%B+%G;%F=%F+%I;%A=%B;%E=%F
 330 LI. #FFE3;G.b
1000a %V=%(%V*100+0.5)/100;R.
2000REM COORDINATE DATA X Y Z
2010 DATA 1,3,1,1,1,1,3,1,1,3,-1,1,1,-1,1,1,-3,1
2020 DATA -1,-3,1,-1,-1,1,-3,-1,1,-3,1,1,-1,1,1,-1,3,1
2030 DATA 1,3,1,1,3,-1,1,1,-1,3,1,-1,3,1,1,3,1,-1
2040 DATA 3,-1,-1,3,-1,1,3,-1,-1,1,-1,-1,1,-3,-1,1,-3,1
2050 DATA 1,-3,-1,-1,-3,-1,-1,-3,1,-1,-3,-1,-1,-1,-1
2060 DATA -3,-1,-1,-3,-1,1,-3,-1,-1,-3,1,-1,-3,1,1
2070 DATA -3,1,-1,-1,1,-1,-1,1,1,-1,1,-1,-1,3,-1
2080 DATA -1,3,1,-1,3,-1,1,3,-1
```

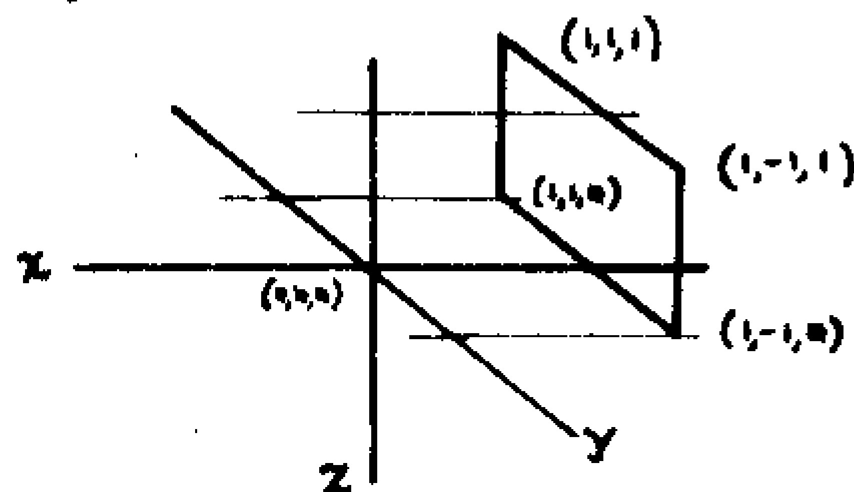
Bovenstaand programma geeft de mogelijkheid om voorwerpen in 3D te tekenen vanuit elke gewenste kijkpositie.
In dit geval tekent het programma een figuur als hieronder:



Het programma berekent de x,y en z-coordinaten t.o.v. de kijkpositie (a.d.h.v. de relatieve coordinaten ten opzichte van het nulpunt, welke in de DATA list zijn opgegeven.).
Aan het begin van het programma moet u de volgende getallen ingeven:

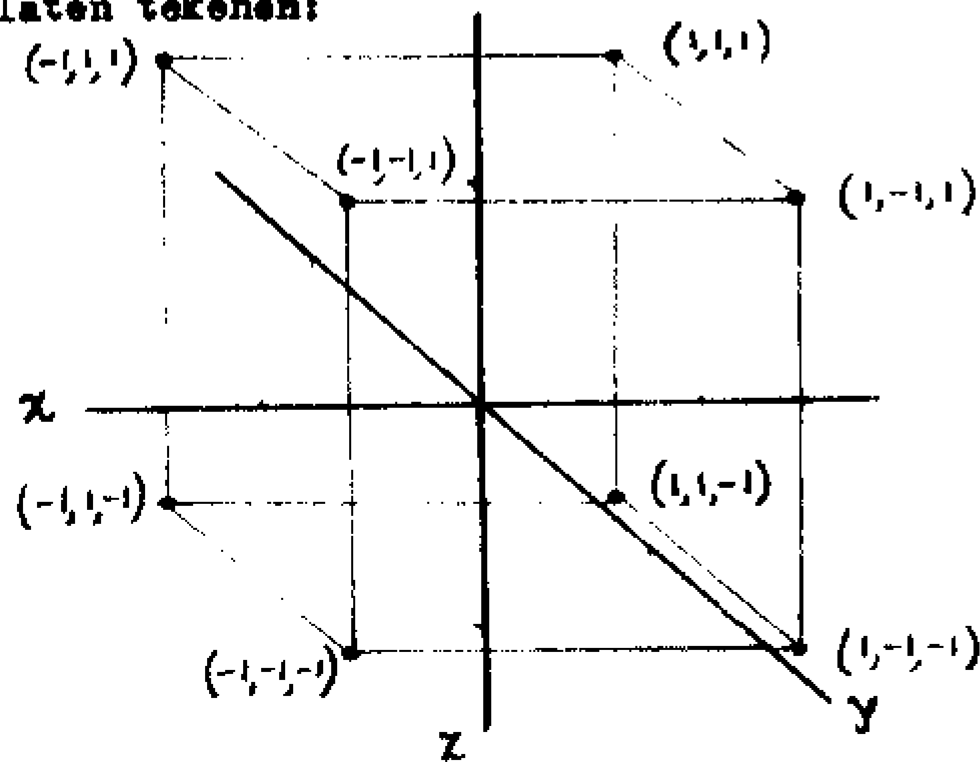
* Azimuth (tussen 0 en 360 graden),d.i. de draaiing van het voorwerp
* Elevatie (tussen 0 en 360 graden),d.i. de helling van het voorwerp
* Afstand (kies hiervoor waarden groter dan 5)
* Azimuth-verandering: nadat een figuur is getekend kan deze door het aanslaan van een toets in een nieuwe positie getekend worden, het programma berekend de nieuwe azimuth en elevatie indien als Azimuth-verandering en Elevatie-verandering getallen groter dan 0 zijn ingegeven.
* Elevatie-verandering, idem
* Vergrotingsfactor, geef in le instantie 10 in, experimenteer met andere waarden

Indien u dit programma andere voorwerpen/figuren wilt laten tekenen dient u slechts de data-list te wijzigen, de list is opgebouwd uit groepjes van 3 data (nl. de x,y en z-coordinaten), de coordinaten geeft u op t.o.v. het nulpunt in het assenstelsel:



Elke nieuwe lijn die getekend moet worden, komt vanuit de laatste coordinaten. Het kan dus voorkomen dat voor verschillende lijnen, een aantal oude lijnen overgetekend moeten worden. Dit gebeurt ook in dit voorbeeld. Het aantal coordinaten-groepen moet in regel 310 aangepast worden aan de nieuwe data-list.

Hieronder nog een voorbeeld hoe men een kubus zou kunnen laten tekenen:



De data-list voor een kubus zou er dus als volgt uitzien:

```
1,1,1,1,-1,1,1,-1,-1,1,1,-1,1,1,1
-1,1,1,-1,-1,1,1,-1,1,1,-1,-1,-1,-1,-1
-1,-1,1,-1,-1,-1,-1,1,-1,-1,1,1,-1,1,-1
1,1,-1
```

Het aantal coordinaten-groepen is nu 16, dus regel 310 verandert in:

```
310 T=T+1; IF T<16 G.d
```

Zo zou je ook bijvoorbeeld de kubus van rubik kunnen vastleggen, en hem vanuit elke gewenste positie bekijken. (Natuurlijk worden dat wel flinke data-lists).

Tenslotte: u kunt dit ook in lagere grafische modes doen, hiervoor moeten regel 270 en 300 aangepast worden. De coordinaten hoeven niet groter opgegeven te worden, omdat u met de vergrotingsfactor e.e.a. kunt regelen. Dit programma maakt gebruik van de toolkit en de FP-ROM. Misschien dat ook iemand dit programma op snelheid kan maken (machinetaal?) zodat je de positie-verandering kunt versnellen en dus "animated graphics" kan verkrijgen.

Eric Snelders, Delfzijl.

Zéér formaatbewust, Erik!

Volgende keer 170x256 mm^2 ??

3000 BAUD!!

Deze routine is op een gewone recorder betrouwbaar en veel sneller dan de Atom routine, ook in het opzoeken van een programma.
Ik heb gekozen voor 3000 Baud en de volgende codering:



Een dergelijk signaal blijkt zelfs voor de zeer slechte recorders die ik heb nog feilloos te verwerken. Omdat de aanloopprocedure voor ieder blok ook compact is, is de routine 16x zo snel als de originele van de Atom.

TOEPASSING

De routine leest of schrijft genummerde blokken van 1/4-K.
Tijdens schrijven en lezen worden de bloknummers geprint; ook die niet tot het gezochte programma behoren. Als men de blokken op een bandje doorlopend nummert is een programma (of data) zeer snel op te zoeken. De routine wordt gewoon met Basic gestart door de vectors te wijzigen (zie programma). Via een eigen interpreter kan natuurlijk ook.

16 K in 55 seconden ....
Een snelle lees/schrijfroutine

SCHRIJVEN (na wijziging van de vector)

SAVE"AAAA BBBB CCCC"
AAAA = beginadres van de te schrijven gegevens
BBBB = nummer eerste blok; dit is vrij te kiezen
CCCC = nummer laatste blok+1
Alle nummers en adressen in hexadecimaal.
De spaties mogen ook een ander teken zijn.

VOORBEELD

Een Basic programma begint op ##2900; P.&TOP geeft 354C; het aantal hele blokken is dus (hex) 36-29=D. Het laatste programma op de band gaat tot 005F (bloknrs., niet bytes). Wil men doorlopende nummering dan gaat het nieuwe programma tot 005F+D=006C de statement wordt dus SAVE"2900 005F 006C". Eerst de recorder starten en daarna de RETURN toets.

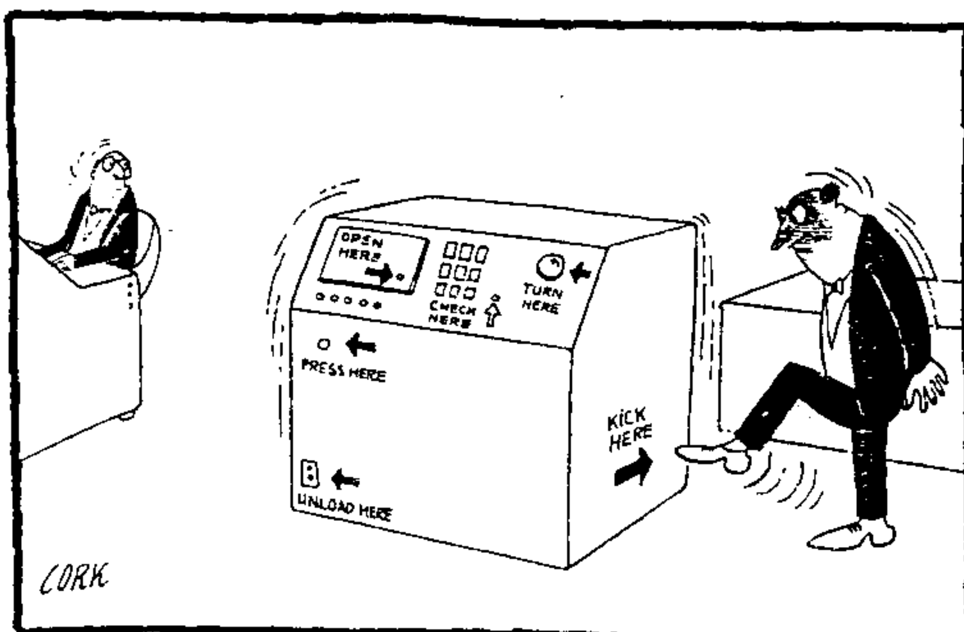
LEZEN (na wijziging van vector)

LOAD"AAAA BBBB CCCC"
De getallen als bij SCHRIJVEN. De routine begint te lezen zodra een blok met nr. BBBB gelezen wordt; nadat bloknummer CCCC gelezen is keert de routine terug naar het aanroepende programma of de monitor. Het lezen kan onderbroken worden met de ESC-toets.
Gebruikt men bloknummer 0000 niet dan kan met LOAD"000000000" snel gecontroleerd worden welke nummers op de band staan (gebruik de REPT-toets; een paar nullen teveel maakt niets uit).
Is op bitniveau iets mis dan geeft de routine de melding drop-out; is de checksum niet in orde komt de melding checksum fout; in beide gevallen stopt het programma.
N.B.: tijdens het lezen met de VDU niet gaan doorschuiven ("scrolling") omdat dit teveel tijd kost, dus bij voorkeur bovenin het beeld werken.
De routine gebruikt de RAM-adressen 165-175 (decimaal).

```
1P.$12"lees/schrijfroutine"'
 2 P."NAAR WELK ADRES MOET"'
 3 IN."GEASSEMBLEERD WORDEN "P
 5 @=4;P."EINDADRES ROUTINE=#"&(P+694
   )';@=8
 10 DIMLL(46)
 11 FORK=0TO46;LL(K)=P;N.
 20 A=P;P."even geduld,aub"'$21
 30 FOR K=1 TO 2;P=A
 35[
 40 JSR LL18  converteer
 50:LL0 LDX@127;LDA@4;STA#B002
 60:LL1 LDA@1;EOR#B002
 70 STA#B002;LDY@63
 80:LL2 DEY;BNE LL2;NOP;DEX
 90 BNE LL1;LDA@1;EOR#B002
100 STA#B002;LDY@31
110:LL3 DEY;BNE LL3;LDA@1
120 EOR#B002;STA#B002;LDY@28
130:LL4 DEY;BNE LL4;LDA170
140 STA 166;JSR LL5 schrijfr.
150 LDA 171;STA 166;JSR LL5
155 LDA@1;EOR#B002;STA#B002
160 LDA 171;JSR#F802 print
170 LDA 170;JSR#F802;LDY@0
180 STY 174;STY 175 checksum
190 LDX@70;STX 167
200:LL11 LDA@1;EOR#B002
210 STA#B002;LDX@63
220:LL12 DEX;BNE LL12
230 DEC 167;BNE LL11
240:LL13 LDA@1;EOR#B002
250 STA#B002;LDX@32
260:LL14 DEX;BNE LL14;EOR@1
270 STA#B002;LDA(168),Y
280 STA 166;CLC;ADC 174
290 STA 174;LDA@0;ADC 175
300 STA 175;LDX@24
310:LL15 DEX;BNE LL15
320 JSR LL5 schrijfr.
330 INY;BNE LL13;LDA@1
340 EOR#B002;STA#B002;LDX@32
350:LL16 DEX;BNE LL16;EOR@1
360 STA#B002;LDX@25
365:LL44 DEX;BNE LL44;CLC;LDA 169
370 ADC@1;STA 169;LDA 174
380 STA 166;LDY 175;JSR LL5
390 STY 166;JSR LL5
395 LDA@1;EOR#B002;STA#B002
397 LDX@25
398:LL45 DEX;BNE LL45
400 CLC;LDA 170;ADC@1
410 STA 170;LDA 171;ADC@0
420 STA 171;CMP 173;BNE LL17
430 LDA 170;CMP 172;BNE LL17
440 RTS einde
450:LL17 JMP LL0 naar begin
500\converteer routine
510:LL18 LDX@0;LDY@0
520:LL19 LDA#140,Y;JSR#F87E
530 ASLA;ASLA;ASLA;ASLA
540 STA 169,X;LDA#141,Y
550 JSR#F87E;ORA 169,X
560 STA 169,X;LDA#142,Y
570 JSR#F87E;ASLA;ASLA
580 ASLA;ASLA;STA 168,X
590 LDA#143,Y;JSR#F87E
600 ORA 168,X;STA 168,X
610 TYA;CLC;ADC@5;TAY;INX;INX
620 CPX@6;BNE LL19;RTS einde
700\schrijfr.8bits
710:LL5 LDX@8;STX 167
720:LL10 LDA@1;EOR#B002
730 STA#B002;LDX@29
740:LL6 DEX;BNE LL6;NOP
750 LSR 166;BCS LL7;NOP
760 NOP;NOP;NOP;BCC LL8
770:LL7 LDA@1;EOR#B002;STA#B002
780:LL8 LDX@29;
790:LL9 DEX;BNE LL9;NOP
800 DEC 167;BNE LL10;RTS eind
810\leesprogr.
820 JSR LL18
830:LL35 LDX@88;STX 166
840 JSR LL20;LDX@5
860:LL36 DEX;BNE LL36;NOP
870 JSR LL27;LDY  166;NOP
880 NOP;JSR LL27;LDA 166
890 JSR#F802;TYA;JSR#F802
900 LDA 166;CMP 171;BNE LL35
910 CPY 170;BNE LL35;LDA 170
920 CLC;ADC@1;STA 170;LDA 171
930 ADC@0;STA 171 startadr+1
940 LDX@8;STX 166;JSR LL20
950 LDY@0;STY174;STY175
955 STY#DC;LDX@3
960:LL37 DEX;BNE LL37 egal.
970:LL38 JSR LL27;LDA 166
980 STA(168),Y;CLC;ADC 174
990 STA 174;LDA 175;ADC@0
1000 STA 175;LDX#B002;STX 165
```

```
1012 LDA#DC;BEQ LL46;RTS
1015:LL46 LDX@12
1022:LL39 DEX;BEQ LL32;LDA 165
1032 EOR#B002;AND@#20
1042 BEQ LL39;LDX@26
1050:LL42 DEX;BEQ LL32
1052 LDA 165;EOR#B002;AND@#20
1070 BNE LL40;LDX@5
1080:LL41 DEX;BNE LL41;INY
1090 BNE LL38;NOP;JSR LL27
1100 LDA 166;CMP 174;BNE LL43
1110 JSR LL27;LDA 166;CMP 175
1120 BNE LL43;LDA 170;CMP 172
1130 BNE LL42;LDA 171;CMP 173
1140 BNE LL42;RTS
1150:LL42 INC 169;JMP LL35
1170:LL43 LDY@1;STY#DC;JSR#F7D1
1180];$P="checksumfout"
1190 ?(P+12)=#EA;?(P+13)=#60
1193 P=P+14
1195[
1196:LL32 STY 164;LDY@1
;STY#DC;JSR#F7D1
;]
1197 $P="drop-out"
1198 !(P+8)=#60EA;P=P+10
1200[\subr.lees"0"+"1"
1210:LL20 JSR#FE71;CPY@59
1220 BNE LL21;BRK
1230:LL21 LDX 166;STX 167
1240:LL22 LDA#B002;TAY;LDX@28
1250:LL23 DEX;BEQ LL20;TYA
1260 EOR#B002;AND@#20
1280 BEQ LL23;CPX@12;BPL LL20
1290 DEC 167;BNE LL22
1300:LL24 LDA#B002;TAY;LDX@28
1310:LL25 DEX;BEQ LL20;TYA
1320 EOR#B002;AND@#20;BEQ LL25
1330 NOP;NOP;NOP;CPX@14;BMI LL24
1340:LL26 TYA;EOR#B002;AND@#20
1350 BNE LL26;RTS
1400\leesr.8bits
1410:LL27 LDX@8;STX 167
1420:LL28 LDA#B002;STA 165;LDX@29
1430:LL29 DEX;BEQ LL32;LDA 165
1440 EOR#B002;AND@#20;BEQ LL29
1450 CPX@18;CLC;BMI LL31;LDX@26
1460:LL30 DEX;BEQ LL32;LDA 165
1470 EOR#B002;AND@#20;BNE LL30
1480 SEC;NOP;NOP
1490:LL31 ROR 166;DEC 167
1500 BNE LL33;RTS
1510:LL33 LDX@4
1520:LL34 DEX;BNE LL34;BEQ LL28
1530];NEXT K;P.$6
1600 P."assembler klaar"''
1610 P."set vectors voor gebruik"'
1619 @=4
1620 P."TYPE:!#20C=#"&A,&(A+#15B);@=8
1630 P.'''"nb:tijdens lezen mag geen "
"scrolling optreden"''
1640 END
>
```

Peter Ehrlich



TOELICHTING

50-90	schrijft als aanloop 127 "0"
-130	schrijft "1"
-155	schrijft bloknr.
-170	bloknummer naar VDU (indien toegestaan)
190-230	schrijft 70 "0": nieuwe aanloop na printroutine
-270	schrijft "1"
-300	haal byte op, tel op bij checksum
-330	schrijf 256 bytes
-355	schrijf "1"
-370	startadres + 256
-390	schrijf checksum
-440	verhoog bloknummer: * = eindnummer, dan klaar
500-620	de converteer routine haalt de gegevens uit de string-buffer en plaatst deze in geschikte vorm in de lokaties 168-173
710	bitteller
-730	fase-om
-740	wachtlus halve bitperiode
750	is het een "1" dan naar 770: extra fase-om
780-790	tweede helft bitperiode
800	alle bits gehad: terug
820	gegevens klaarzetten
-840	subroutine: lees 88 "0" en wacht op "1"
-890	lees en naar VDU bloknummer
-910	niet het gezochte: terug naar begin
940	lees 8 "0" en wacht op "1"
970	lees 8 bits, werk checksum bij
1010	goed gelezen (statusbyte): doorgaan
-1070	lees "1": niet goed: drop-out melding
-1090	blok klaar
-1120	lees en controleer checksum
-1140	laatste bloknummer: klaar
-1198	foutmelders
-1220	escape?
-1280	meet periodetijd: te lang of te kort: opnieuw beginnen
1290	genoeg "0" gelezen?
-1350	wacht op "1"
1410	bitteller
-1440	wacht op fase-om: te lang geduurd: drop-out
1450	korte periode: is "1", wacht op tweede fase-om anders "0", meteen door naar 1490
1490	bit inschuiven m.b.v. carry-bit
-1520	volgende bit of klaar

adressen:

a4	fout-index (positie in blok)
a5-a7	hulp-velden
a8,a9	startadres
aa,ab	start bloknummer
ac,ad	eind bloknummer
ae,af	checksum
dc	fout-status (OK=0)

RUISVRIJ PLOTTEN

Jan Wijnen

Als men iets verandert in het beeldgeheugen, dan gaat dat gepaard met ruis op de t.v. Dit wordt veroorzaakt door het onderbreken van de VDU door de CPU. Elke keer als er iets veranderd wordt, moet de VDU even stoppen met het genereren van het televisiesignaal.
Je hebt hier twee oplossingen voor: De eerste is dat je alleen maar iets verandert als de VDU in de terugslag zit van rechtsonder in het beeld naar linksboven. Dit kun je doen door bijv. het toevoegen van WAIT-statements.
De tweede oplossing is het synchroniseren van de CPU met de VDU. Dit houdt in, dat je de klok van de CPU en de VDU van eenzelfde bron betrekt. Als dan de CPU het beeldgeheugen gebruikt, dan heeft de VDU het niet nodig en omgekeerd.
In de ATOM zitten twee klokgeneratoren, die van de VDU en die van de CPU. In verband met de beeldfrequentie is het niet mogelijk om de CPU-klok aan de VDU te koppelen.

Oplossing: Haal IC 44 uit zijn voetje en buig pootje 13 naar buiten. Soldeer dit met een draadje aan de middenpen van de omschakelaar.
Soldeer de ene kant van de schakelaar aan pootje 2 van IC 45 en de andere kant aan pootje 10 van IC 9.

Je kunt nu omschakelen tussen twee klokfrequenties voor de CPU. Dit is nodig omdat de klokfrequentie van de VDU niet 4 MHz is maar 3.58 MHz. Dit heeft tot gevolg, dat de CPU zijn werk 10% langzamer doet. En dat is vervelend in tijdgevoelige kwesties als SAVE- en LOAD-operaties.
Het kan zijn dat, als je omschakelt naar 3.58 MHz, de ruis niet verdwijnt maar de processor toch langzamer werkt. Dit komt omdat de CPU niet op 4 MHz werkt, maar het kloksignaal nog door 4 deelt. De CPU loopt dus wel synchroon met de VDU maar niet in fase. Door nu een paar keer om te schakelen, komt het zaakje op een gegeven ogenblik wel in fase en dan ben je de ruis wel kwijt.
Zelf heb ik de omschakelaar achter op het board gelijmd op de plaats waar normaal de koelplaat zit. Het knopje steekt naar buiten, zodat omschakelen geen probleem is.

onderdelen:

S1 omschakelaar

A = slow, geen ruis
B = fast, wel ruis

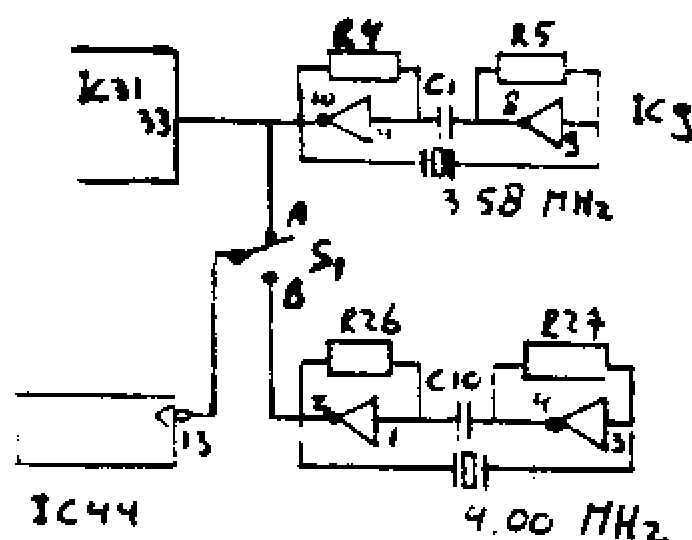


fig.1.
ruisvrij beeldscherm

GEEN VRAGEN!!!

Deze korte routine zorgt ervoor dat het IN. statement gebruikt kan worden zonder dat er een vraagteken verschijnt.

```
10 REM NO ???
20 P=#80;[;LDA @82;STA #208;LDA@254;STA #209;RTS;];Z=#FE940080
30 REM DEMONSTRATE USE
40 PRINT "ENTER AMOUNT:"; !#208=Z;INPUT A
50 END
```

BRON: YOUR COMPUTER augustus 1982

HET NIEUWE BASICODE - PROTOCOL   Rudi van Drunen.

Bij basicode 1 was het zo dat er altijd veel aangepast moest worden, dit aanpassen hoopt men met BASICODE-2 terug te brengen. dit wil men bereiken met het schrijven van subroutines voor de meest voorkomende machine-afhankelijke statements;deze subroutines worden na het inlezen van het basicode programma hieraan toegevoegd. Het eigenlijke programma begint pas op regel 1000; van regel 100 tot 1000 staan de subroutines voor de machine-afh. statements. NB op r. 10 moet dus altijd staan: GOTO 1000.
Nu kan een basicode 2 uitgezonden programma ook met het basi-code-1 vertaalprogeamma van Arie Marchel binnen gehaald worden maar dan moeten de subroutines er zelf bij geschreven worden.

Een overzicht van de subroutines:

-R. 100: sbr. voor het wissen van het scherm; bij ons dus:
100 P.$12;RETURN

-R.110 : sbr. voor het positioneren van de cursor;plaats 0,0 is hierbij links-boven op het scherm; de plaatsen waarheen de cursor moet gaan in de variabelen VE(vertikaal) en HO(horizontaal)

-R. 120: deze sbr. doet het omgekeerde van de vorige: in de variabelen VE en HO komt te staan waar de cursor op het moment van het aanroepen staat.

-R. 200: sbr. die kijkt of er een toets ingedrukt was, zo ja, dan komt in IN$ te staan welke toets het was;zo nee, dan blijft IN$ leeg.

-R. 210: deze sbr. doet ongeveer hetzelfde als de vorige,maar wacht net zolang tot er een toets ingedrukt wordt.

-R. 250: deze sbr. laat de computer een piepje geven.

-R. 260: Door deze sbr. wordt een random getal gegenereerd, en neergezet in de variabele RV de kleinste waarde van het getal is 0; maar het mag niet groter of gelijk zijn aan 1.

-R. 270:deze sbr. geeft in FR het aantal nog vrije bytes.

-R. 300:deze sbr. geeft STR$(SR) in SR$

-R. 310:het getal dat in SR staat wordt omgezet in SR$ (door sbr. R. 300), de lengte van SR$ is na afloop gelijk aan wat u in CT hat opgegeven;is CN groter 0,dan zal SR$ een deci-male punt bevatten; met daarna nog CN cijfers.
Een voorbeeld:
SR=123.456;CT=7;CN=2;GOSUB 310
Dit levert; SR$=' 123.46'

-R. 350; deze sbr. zorgt ervoor dat SR$ geprint wordt op een printer, hierna wordt een 'linefeed gegeven naar de printer.

-R. 360 :geeft 'carriage return' en 'linefeed' naar de printer

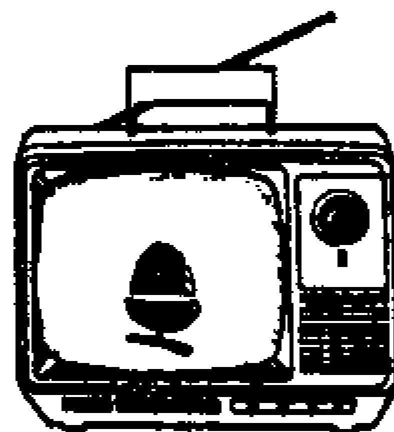
Deze info. heb ik uit de op 09 jan uitgezonden uitleg prgr'S. Verdere informatie: NOS hobbyskoop, Postbus 1200,1200 BE, Hilversum.

ALS MEN NA DE GRMOD-OPDRACHT CLEAR 1 GEEFT, DAN
KRIJGT MEN DE DAARNA TE PRINTEN TEKST, IN DE BETREFFENDE GRAPHICSMODE.
EEN MOOIE GROTE GESTIPPELDE LETTER.
DE GROTE VAN DE LETTER IS AAN TE PASSEN, DOOR RESP. CLEAR 1,2,3,4,
UITSTEKEND GESCHIKT OM MAKKELIJK MEDEDELINGEN OF RECLAME BOODSCHAPPEN
TE DOEN. OOK GESCHIKT VOOR B.V. GROTE CIJFERS DIGITALE KLOK.

VOORBEELD:
```
10 GRMOD
20 ?#E1=0
30 CLEAR 1
40 P.'"     ACORN"'
50 END
```

BIJ ONDERSTAAND PROGRAMMA KRIJGT U HET EMBLEEM, VAN DE REGIO, WAAR HET HEEL GOED TOEVEN IS.

```
10 REM CLUBEMBLEEM REGIO BRABANT-OOST
20 REM GEBRUIK JOSBOX
30 REM TINY VERSCHUREN
40 GRMOD
50 ?#E1=0
60 CLEAR 2
70 P.'"          ACORN"'
80 P."  COMPUTERCLUB"'
90 P." ""BRABANT-OOST"""'
100 Y=95
110 RESTORE
120[illegible]Y=Y-1
130 IF Y=68 THEN Y=67
140 READ X
150 IF X=100 T.G.[illegible]
160 PLOT 4,X,Y
170 READ X
180 IF X=100 T.G.[illegible]
190 PLOT 5,X,Y
200 IF Y=53 T.G.[illegible]
210 G.A
220[illegible]READ X
230 IF X=100 T.G.[illegible]
240 PLOT 4,X,Y
250 READ X
260 IF X=100 T.G.[illegible]
270 PLOT 5,X,Y
280 READ X
290 IF X=100 T.G.[illegible]
300 PLOT 4,X,Y
310 READ X
320 IF X=100 T.G.[illegible]
330 PLOT 5,X,Y
340 Y=Y-1
350 G.[illegible]
360[illegible]END
370 DATA 18,22,16,24,15,25,14,26,13,
         27,12,28,11,29,11,29,10,30
380 DATA 9,31,8,32,8,32,7,33,7,33,6,
         34,6,34,6,34,5,35,5,35,5
390 DATA 35,4,36,4,36,4,36,3,37,3,
         37,3,37,3,37,3,37,4,36,4,36
400 DATA 5,35,5,35,6,34,6,34,7,33,7,
         33,8,32,9,31,11,29,13,27
410 DATA 15,25,17,23,16,24,19,21,19,
         21,19,21,5,7,19,21,4,9
420 DATA 19,21,6,14,19,21,8,22,19,26
         ,10,24,16,28,16,28,21,34
430 DATA 21,34,25,30,25,30,100
```



```
1Ø  REM Met deze routine worden de spaties óók
2Ø  REM wit als U inverse video gebruikt.
3Ø
4Ø  P=#28AØ;[PHA;LDA#BØØ1;CMP@#7F;BEQ#28B3;LDA#E7
5Ø  CMP@#6Ø;BEQ#28B3;PLA;JSR#FE52;RTS;PLA;CMP@#2Ø
6Ø  BEQ#28BC;JSR#FE52;RTS;LDA@#FF;JSR#FE52;RTS;]
7Ø  ?#2Ø8=#AØ;?#2Ø9=#28;E.
```

Frank de Groot

```
 10REM TEKENING IN MODE 0
 20REM TEKENT IN GRIJS IPV WIT
 30REM L.BIJNAGTE
 40DIM XX40,YY40,LL0
 50P.$12"busy"
 60FOR I=0 TO 36
 70XXI=%(24*SIN(RADI*10)+32)
 80YYI=%(24*COS(RADI*10)+24)
 90N.
100DO
110GOSUB 200
120FOR N=0 TO 100;WAIT;NEXT N
130GOSUB 220
140FOR N=0 TO 100;WAIT;NEXT N
150UNTIL 0
160FORI=0TO 36
170IFI=0MOVEXX0,YY0
180DRAWXXI,YYI
190NEXT;RETURN
200FOR N=#8000TO#81FF;?N=#C0;NEXT
210GOTO 160
220CLEAR0
230GOTO 160
240REM REGEL 200 KAN IN MACHINETAAL MET
250P=#28A0;[LDY@0;LDA@#C0;:LL0 STA#8000,Y
260STA#8100,Y;INY;BNE LL0;RTS;]
270REM EN IS AAN TE ROEPEN MET LINK #28A0
280REM DUS VOOR DE MOVE,PLOT EN DRAW COMMANDO'S
290REM BIT3 VAN #B002 MOET LAAG ZIJN, ANDERS WERKT HET NIET
300END
```



40x24 VDU-PK

Geeft een beeldscherm van 24 regels en 40 kolommen.
Upper en Lowercase (grote en kleine letters) plus geinverteerd.
Vertikaal instelbaar tussen 1 en 24 regels, vanaf de bovenkant van het scherm (?V=Aantal regels).
Horizontaal instelbaar tussen4 en 40 kolommen in 4-voud oplopend vanaf de linkerkant van het scherm (?H=3/4xAantal kolommen).
Naar keuze zwarte letters op wit beeld of witte letters op zwart beeld(?F=0 of #80 voor resp. zwart op wit of wit op zwart).
Karakters met bit 7=1 worden geinverteerd weergegeven, intoetsen na CTRL-6.
CTRL-3 en 4 geven resp. het normale 32 kar. VDU en het nieuwe 40 kar. VDU (P.$#13 en P.$#14)
Met CTRL-2 kan tussen VDU 32 en 40 gewisseld worden zonder dat het scherm gewist wordt.
CTRL-1 print de inhoud van de input buffer (datgene wat je hebt ingetypt tot de eerste return).
Regelinvoer lengte is verhoogd van 64 tot 192.
Auto repeat op alle toetsen met super snelle repeat via Rept toets. Shift en CTRL tegelijkertijd ingedrukt levert vertraagd afdrukken. Dit programma is overal te plaatsen: lengte 2K (1K8 om precies te zijn).
Probeer ook eens: ?H=15; ?V=12; ?#B000=#B0 en type maar wat in.. (20x12VDU, Leuk !!!!!).
Het programma gaat naar het bandjesarchief.

Paul Kuyper

*** FORMAT 12/7/'83 ***

```
  0   REM FORMAT 12/7/'83
 10 i REM INITIALISATIE
 20   DIM LL7:FOR I=0 TO 7;LLI=#FFFF NEXT I
 30   DIM C0
 40   Q=#FE52:D=#5F:P.$21
 50   FOR I=1 TO 2:P=C
 60 [
 70  LL0
 80   LDA@0:STA#0:TAY
 90   LDA@32;JSRQ
100   LDA(#60),Y:TAX
110   CMP@58:BEQ LL3
120   CMP@91 BEQ LL3
130   CMP@93:BEQ LL3
140   CMP@#60:BPL LL1
150   LDX@32:DEY
160  LL1
170   INY:TXA:JSRQ
180   LDA@32;JSRQ
190  LL2
200   LDA(#60),Y:TAX
210  :LL3
220   INY
230   CMP@13:BNE LL4
240   JSR#CD54:INCD
250   RTS
260  :LL4
270   TXA;BMI LL6
280   CMP@31;BPL LL6
290  LL5
300   LDX@31
310  LL6
320   TXA:JSRQ
330   INC#0:LDA#0:CMP@#50:BMI LL2
340   JSR#CD54
350   INCD;LDA@32 LDX@10
360  .LL7
370   JSRQ;DEX;BNE LL7
380   LDA@0:STA#0:JMP LL2
390 ]
400   NEXT I
410 h REM HOOFDPROGRAMMA
420   P.$6$12:?D=5:@=5
430   DO IN."  TEKSTRUIMTE "A A=A*256:E=A,UNTIL A<#FFFF
440   IF ?A<>13 OR A?1>127 P."LEGE TEKSTRUIMTE"';END
450   P.' "  TOETS ALS PRINTER STARTKLAAR"';LI.#FFE3
460   A=A+1
470   P.$21$2$27$48$27$61
480   P.$27$14"*** "$(A+6)"  ***"'''$27$15
490   DO B=A+LEN(A+2)+3,P."  "?A*256+A?1;!#60=A+2;LI.C A=B
500   IF ?D>66 P.''''':?D=0
510   UNTIL ?B>127
520   @=0:P.''"               PROGRAMMALENGTE "B+1-E" BYTES"' ?D=?D+3
530   IF ?D<72 DO P.' ?D=?D+1 UNTIL ?D=72 OR ?#B001=127
540   P.$3$6:@=8 E.


      PROGRAMMALENGTE 1068 BYTES
```

```
LIST
   0REM FORMAT 12/7/'83
  10iREM INITIALISATIE
  20DIM LL7:FOR I=0 TO 7,LLI=#FFFF NEXT I
  30DIM C0
  40Q=#FE52:D=#5F:P.$21
  50FOR I=1 TO 2.P=C
  60[
  70:LL0
  80LDA@0:STA#0:TAY
  90LDA@32;JSRQ
 100LDA(#60),Y:TAX
 110CMP@58:BEQ LL3
 120CMP@91,BEQ LL3
 130CMP@93;BEQ LL3
 140CMP@#60;BPL LL1
 150LDX@32;DEY
 160:LL1
 170INY;TXA;JSRQ
 180LDA@32;JSRQ
 190:LL2
 200LDA(#60),Y;TAX
 210:LL3
 220INY
 230CMP@13 BNE LL4
 240JSR#CD54 INCD
 ( -- ETC )
```

Ziet u het VERSCHIL ?

Wat zal het toch MOOI zijn in Acorn Nieuws met al die FORMATs!



```
0   REM SNELLE SPIEGEL SCHERM : 9/7/'83
10  DIMLL4;F.I=0TO4;LLI=-1;N.
20  DIM C0
30  F.I=1TO2;P=C
40 [
50 :LL0
60  LDX@0;STX#80
70  LDA@#80;STA#81
80 :LL1
90  LDY@0
100 :LL2
110  LDA(#80),Y;STA#82;BEQ LL3
120  ROLA;ROR#82
130  ROLA;ROR#82
140  ROLA;ROR#82
150  ROLA;ROR#82
160  ROLA;ROR#82
170  ROLA;ROR#82
180  ROLA;ROR#82
190  ROLA;ROR#82
200 :LL3
210  TYA;TAX;STA#0;SEC;LDA@31;SBC#0;TAY
220  LDA(#80),Y;STA#83;BEQ LL4
230  ROLA;ROR#83
240  ROLA;ROR#83
250  ROLA;ROR#83
260  ROLA;ROR#83
270  ROLA;ROR#83
280  ROLA;ROR#83
290  ROLA;ROR#83
300  ROLA;ROR#83
310 :LL4
320  LDA#82;STA(#80),Y
330  TXA;TAY
340  LDA#83;STA(#80),Y
350  INY;CPY@16;BNE LL2
360  LDA@32;CLC;ADC#80;STA#80;LDA@0;ADC#81;STA#81
370  CMP@#98;BNE LL1
380  RTS
390 ]
400  N.
410  E.

     PROGRAMMALENGTE 644 BYTES
```

In dit zoekplaatje staat het programma van Rob van Dort links. Dat van Bram Poot 66k, als u de pagina op z'n kop zet. Draait u de pagina weer normaal, dan vindt u onderaan een hogere versnelling voor Bram's programma.

De routine gebruikt de ZP-geheugens #8-#C (RND-ruimte) en wordt aan geroepen met LINK SS0. Het kan overigens nog sneller door een beetje met instructies te schuiven, zodat PHA & PLA vervangen kunnen worden door TAX & TXA (±9 msec sneller):

```
160       LDY#8;LDA(#A),Y;STA#C;LDY@8
170:SS3  ROL#C;RORA;DEY;BNESS3
180       TAX;LDY#9;LDA(#A),Y;STA#C;LDY@8
190:SS4  ROR#C;ROLA;DEY;BNESS4
200       LDY#8;STA(#A),Y;LDY#9;TXA;STA(#A),Y
```

Groetn.

(Bram Poot)

```
10 REM SPIEGELEN
20'Een andere methode om het
30'beeldscherm te spiegelen,
40'n.a.v. een programma van
50'Rob van Dort
60'De routine duurt slechts
70' 0.85 sec.
80
90 DIM SS5;P.$21
100 FOR I=1 TO 2;DIM P-1
110[
120:SS0 LDA@#98;STA#B;LDY@0;STY#A
130:SS1 STY#8
140:SS2 LDA@31;SEC;SBC#8;STA#9
150      CMP#8;BCCSS5
160      LDX@8;LDY#8;LDA(#A),Y;STA#C
170:SS3 ROL#C;RORA;DEX;BNESS3
180      PHA;LDX@8;LDY#9;LDA(#A),Y;STA#C
190:SS4 ROR#C;ROLA;DEX;BNESS4
200      LDY#8;STA(#A),Y;LDY#9;PLA;STA(#A),Y
210      INC#8;BNESS2
220:SS5 LDY@0
230      LDA@#20;CLC;ADC#A;STA#A;BNESS1
240      DEC#B;BMISS1;RTS
250]P.$6;N.I
260 END
```

Bij de programma's

FORMAT : kort gezegd doet het programma zoals is te zien aan de beide listings de compacte,afgebroken listing is geproduceerd door de standaard opdracht "LIST", de volledige listing door "FORMAT".
Het programma verwacht op de eerste regel van het te formatteren programma een "REM" met daarachter de naam van het programma. Deze naam wordt op dubbele breedte afgedrukt als kop.
FORMAT verwacht kettingformulieren met bladen van A4-lengte. lange listings worden ter hoogte van de naad in het papier onderbroken zodat niet op de naad geprint wordt.
Drie spaties worden geprint tussen het regelnummer en het begin van de BASIC-tekst. Een BASIC-label komt op de plaats van de middelste spatie,evenals de assembler open- en sluithaken, mits deze aan het begin van een regel staan.
Assembler labels aan het begin van een regel worden twee posities naar links geprint; wilt u alleen de dubbele punt naar voren hebben,verander dan 113 in regel 110 in 111.
Regels langer dan 80 tekens worden afgebroken en op een volgende regel netjes verder afgedrukt (dit aantal tekens is te veranderen door #50 in regel 330 aan te passen).
Grafische- en contolcharacters worden vervangen door een $31 ($31 geeft op mijn printer een blokje op het papier,zodat ik weet dat er iets aan de hand is. Mocht uw printer $31 niet slikken, maak er dan bijv. $32 (=spatie) van); als uw printer de grafische characters wel kan verwerken, laat dan de opdracht 'BMI LL6' in regel 270 vervallen.
Aan het einde van het geprinte programma wordt de lengte hiervan in bytes afgedrukt.
Het papier wordt doorgespoeld tot aan de eerstvolgende papiernaad. Dit spoelen is te onderbreken met de SHIFT-toets.

Printer specifieke codes:
regel 300 : $31=print unidentified character
regel 470 : $27$48=reset printer
            $27$61=kies 12.5 char/inch
regel 480 : $27$14=dubbelbrede characters
            $27$15=normale characters

Het spreekt natuurlijk voor zich dat u dit programma in een andere tekstruimte moet onderbrengen en runnen dan het programma dat u uit wilt laten printen.
Als u dit programma overtikt moet u de spaties na de regelnummers niet invoeren!!

SNELLE SPIEGEL SCHERM : dit programma doet precies hetzelfde als het programma dat ik in de vorige A.N. publiceerde, alleen zo'n 50* sneller.
Je staat op die manier als programmeur wel even in je hemd, maarzoiets kan iedereen gebeuren.....
Binnen 0,4 sec. na het intoetsen van LINK LL0 is uw grafische scherm CLEAR4 gespiegeld.

Nogmaals : de extra VIA('s).

Vindt u het ook zo weinig : 1 poort die vrij te gebruiken is ?
Wat dacht u hiervan :

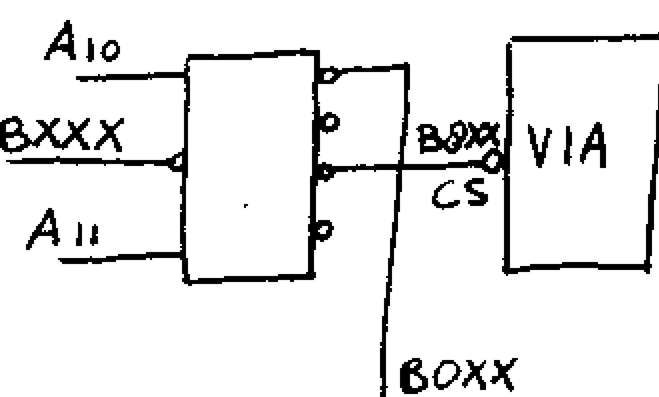
* een altijd gelijklopende Cmos-klok.
  Dit kost u : 2 poorten.
  Totaal vrij te gebruiken : 1 poort.
* aansluiting voor bijvoorbeeld
  - A/D-D/A converter
  - 2 of 3-dimensionale digitizer met hoge resolutie
  - modelbaan besturing
  - ????

  Dit kost u : 2 poorten.
  Totaal vrij te gebruiken : 3 poorten.
* Niet genoeg ? 1 VIA extra geeft 2 poorten extra
  Totaal vrij te gebruiken : 5 poorten.

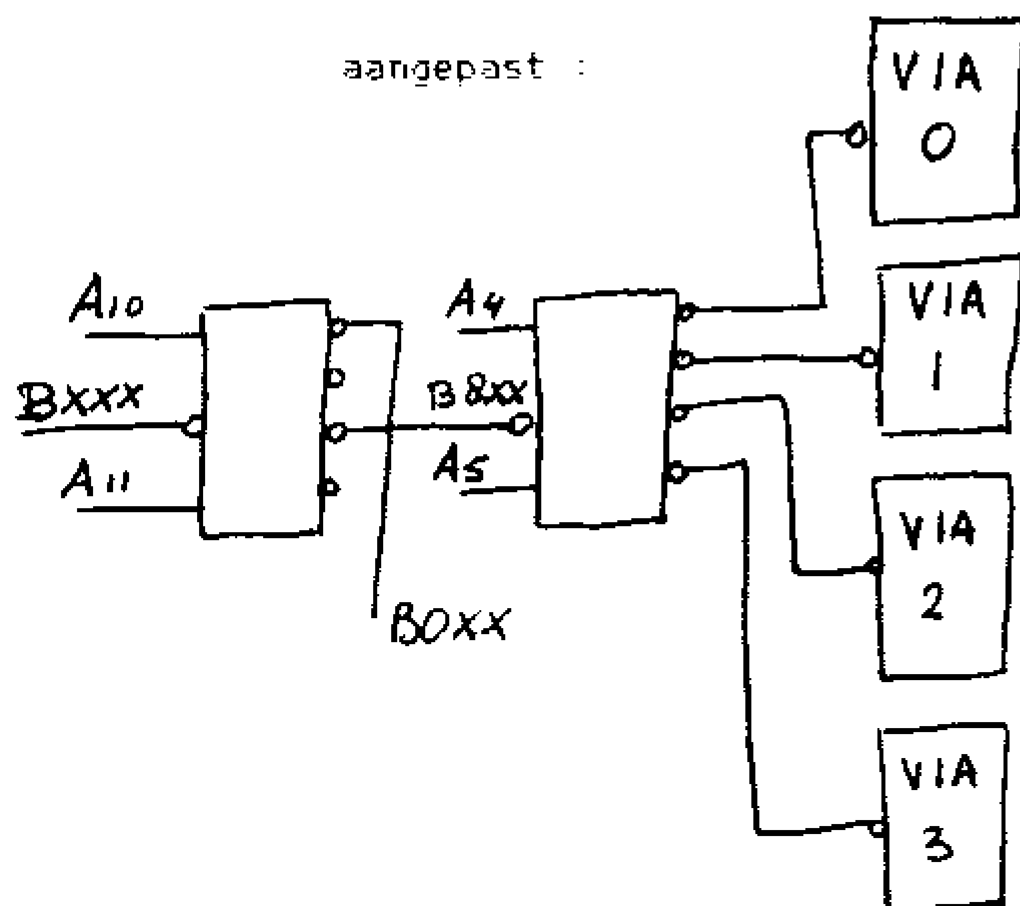
Financieel offer : FL 2,50 + FL 15,= per VIA (via HCC).
Waar plaatsen we deze VIA's in ons geheugen ?
Wel, onze standaard VIA reserveert 1024 bytes (#B800-#BBFF) voor zich, waarvan er maar 16 gebruikt worden.
Niet netjes, en daar gaan we wat aan doen.
Door toevoeging van het IC 74LS139 op de juiste plaats kunnen in hetzelfde geheugengebied 3 extra VIA's een plaatsje vinden (*)

Schema's :

standaard :



aangepast :



Bouwbeschrijving :

- Haal IC 49 uit zijn voet en buig pin 6 naar buiten
- haal IC 1 uit zijn voet en buig pin 23 naar buiten
- soldeer de 74LS139 alleen met pin 8 en pin 16 vast op IC 49 (dit is ook een 74LS139)
- soldeer A4 (pin 15 IC 2) aan pin 2 van de '139
- soldeer A5 (pin 13 IC 2) aan pin 3 van de '139

- soldeer pin 1 van de '139 aan pin 6 van IC 49
- soldeer CS (pin 23) van VIA 0 (onze standaard-VIA) aan pin 4 van de 139
- idem van VIA 1 aan pin 5
- idem van VIA 2 aan pin 6
- idem van VIA 3 aan pin 7

Van de 74LS139 (een 4 uit 2 decoder) is nu slechts de helft gebruikt; u kunt theoretisch op dezelfde manier als boven beschreven 3 VIA's plakken boven de PIA (IC 25)

(x) bij mij werkt de schakeling met 1 extra VIA feilloos; of de printsporen de belasting van nog eens twee VIA's aankunnen moet praktisch worden beproefd (komt binnenkort!)

OPTILLUSIE : zoals vorige keer beloofd het programmaatje waarmee de diepte-suggestie is op te wekken.
Het principe is eenvoudig :

1 genereer een willekeurig stippenpatroon
2 spiegel dit patroon, met dien verstande dat alle punten binnen een geometrische figuur 1 positie naar links (of rechts) worden geschoven.
Op deze manier wordt 1 verdieping verkregen; doen we deze truc nogmaals met een figuur binnen deze figuur dan verkrijgen we nog een verdieping.
In mijn programmaatje worden zo vier vierkanten gemaakt.
Het puntenveld heeft een dichtheid van ca. 50 %. Zorg wel dat deze dichtheid niet beneden 35% komt, omdat anders het grafisch gedeelte van onze hersenen niet genoeg aangrijpingspunten heeft om de afzonderlijke figuren te herkennen.
Het beste resultaat verkrijgt u als u van uw grafische scherm een dump op dubbele breedte laat maken.

U ziet, mijn persoonlijke voorkeur gaat sterk uit naar wat serieuzer grafisch werk. Volgende keer hoop ik iets te kunnen laten zien van een grafisch veld van 512*768 punten (=48kByte!).

Zou onze vereniging niet in aanmerking komen voor koninklijke goedkeuring ?



met vriendelijke hobbygroet,

Rob van Dort

Communicatie tussen twee Acorn Atom's.

Er is mij gevraagd een artikeltje te schrijven over het verbinden van twee Acorn's via de cassette poort. Hier is het dus.

De grap is dat wat op de ene Acorn ingetypt wordt ook op het scherm van de andere Acorn verschijnt, zelfs control-code , zoals ctrl-l, gebeurt ook op het ontvangende scherm.
Stel u woont in een flatgebouw en u heeft een medebewoner die ook een Acorn heeft dan kunt u met elkaar communiceren via de Atom.
Wat is hier voor nodig??
Weinig! Het enigste wat u nodig heeft is een cassette-snoer van de lengte die tussen de beide Acorns is, en een programmaatje.
Nu is dat natuurlijk niet zonder méér een cassette-snoertje zoals u die in de winkel koopt, nee, u moet er zelf iets aan veranderen.
Wat u moet doen: in de eerste plaats moet u ervoor zorgen dat, als u gebruik maakt van een 5-polige stekker, pootje 1&2 en pootje 4&5 niet aan elkaar zitten!! U moet slechts gebruik maken van de buitenste pootjes waarbij de min of ook wel aarde natuurlijk blijft zoals het is.
Daarna moet u bij het ene stekkertje het draadje van pootje 1 naar 5 leggen en het draadje van pootje 5 naar pootje 1.
M.A.W.: gewoon links en rechts verwisselen zodat straks de uitgang van de cassette-poort van de ene Acorn aan de ingang van de cassette-poort van de andere Acorn komt en andersom.
Als u dit voorelkaar heeft, dan is de verbinding tussen de twee Acorns gemaakt. Nu dus alleen nog het programma.

```
10 P."zenden"'
20 WAIT;WAIT;WAIT;WAIT;
30 KEY C;IF C=0 G.30                    KEY is een Toolkit commando.
40 BPUT A,C
50 IF C=CH"Q";P.';G.100
60 P.$C
70 G.20
100 P."ontvangen"'
110 C=BGETC;P.$C
120 IF C=CH"Q";P.';G.10
130 G.110
```

Uitleg: De beide Acorns hebben aan dit programma genoeg, voor beide Acorns hetzelfde programma dus.
Men typt op de ene Acorn RUN in en op de andere Acorn GOTO 100.
Nu verschijnt op het scherm van de ene Acorn "ontvangen" en op het scherm van de andere Acorn verschijnt de text "zenden".
Als men nu op de "zend-Acorn" een zin of een woord intoetst dan zal dat woord op de andere Acorn verschijnen.
De ontvang-Acorn kan niet zenden en moet rustig kijken wat er op het scherm komt totdat....... de zend-Acorn een "Q" wegstuurt die de zend-Acorn naar ONTVANGEN en de ontvang-Acorn naar ZENDEN zet.
U kunt natuurlijk voor de "Q" ook SHIFT-Q gebruiken of een andere toets.

W. Schreuder (regio Noord)

MORSE DECODER

Het morse programma dat in Acorn nieuws 3 staat, gebruikt slechts 8 regels van het beeldscherm. De hier beschreven verandering aanvulling, zorgt ervoor dat het gehele scherm wordt benut.

```
41   L=#8A:?L=1
1600:BB40 LDX L
1605 BNE BB45
1610 LDX K
1620 STA #8000,X
1630:BB46 CPX @#FF
1640 BEQ BB41
1645 INX
1647 STX K
1650 RTS
1651:BB45 LDX K
1652 STA #8100,X
1653 JMP BB46
1660:BB41 LDA @0
1665 STA K
1667 LDX L
1668 CPX @0
1669 BNE BB47
1670 LDA @#FF
1671 STA L
1672 RTS
1675:BB47 STA L
1676 RTS
```

Hans Brouwer (PE1FJA)

RECORDER DIAGNOSE PROGRAMMA

Peter Ehrlich

Het programma vraagt signaal of frequentie.
Signaal geeft monsters van het uitgangssignaal van de recorder zoals de computer dit ziet. Dit signaal moet vrij zijn van snel herhalende faseovergangen (naalden) en een redelijk symmetrische blokgolf laten zien. Is dit niet het geval dan kan men een laagdoorlaatfilter (zie A.N.) proberen.
Het frequentie programma geeft de gemiddelde frequentie van de hoge en lage tonen (1/periodetijd). Dit ter controle van de snelheid van de recorder. Het is alleen betrouwbaar als het signaal behoorlijk is, dus eerst controleren met het signaalprogramma.

N.B.: Het programma moet precies als de listing ingetypt worden; als men dit bezwaarlijk vindt moet men regel 10 veranderen van:
...,P(-1) wordt: ...;P=#2EE2.

```
 1REMrecorder diagnose
 10 DIMB(255),LL(6),P(-1)
 15 FORI=0TO6;LLI=P
 20 P.$21;GOS.a;P.$6$12
 30 IN."SIGNAAL OF FREKWENTIE "$B
 40 P.''"PLAY TAPE"'
 50 IF ?B=CH"S";G.s
100 IN."METEN ROND FREKWENTIE (BV 2400
) "F
110 P.$12,"FREKWENTIE:";@=5
120 DO LINK LL0
130 T=0;S=0;E=(500000/F-37)/17+1
140 L=7*E/10;H=10*E/7
150 FORK=0 TO 255
160 IF B?K>L;IF B?K<H;S=S+B?K;T=T+1
170 NEXT K
175 IFT<40;P."       "$8$8$8$8$8;G.200
180 G=500000*T/(17*S+20*T)
190 P.G,$8$8$8$8$8
200 UNTIL0
300sP."DRUK OP TOETS VOOR MONSTER"'
305 FORK=1TO60;WAIT;N.
310 CLEAR4
320 FOR K=10 TO 180 S.20
330 LINK#FFE3;LINK LL5
340 MOVE(-1),K
350 FORI=0 TO 255
355 J=B?I+K
360 DRAWI,J;MOVE(I+1),J
370 N.;N.;END
400a[\TIJDMETING EN SAMPLE ROUT
405:LL0 LDX@0
410:LL1 LDA#B002;STA#80
420:LL2 LDA#80;EOR#B002;AND@#20
430 BEQ LL2 WACHT OP FASE-OM
440:LL3 LDY@0;LDA#B002;STA#80
450:LL4 INY;BEQ LL1;LDA#80
460 EOR#B002;AND@#20;BEQ LL4
470 TYA;STA B,X;INX;BNE LL3;RTS
500:LL5 LDX@0
510:LL6 LDA#B002;AND@#20;LSRA
520 LSRA;STA B,X;INX;BNE LL6
530 RTS;];R.
```



AFTREKKEN VAN 2 GETALLEN MET EEN ZEER KLEIN VERSCHIL
==

PROBEER HET VOLGENDE **VOORBEELD:**

FP. 1-.999968

HET ANTWOORD IS IN ZOVERRE FOUT, DAT HET NIET GOED IS!!
DE JUISTE UITKOMST IS NATUURLIJK 3.2E-5.

WAT HIER VERTOOND WORDT IS EEN ERNSTIGE TEKORTKOMING, DIE IN VOORKOMENDE GEVALLEN DE OPLOSSING VAN BEPAALDE PROBLEMEN, DIE WISKUNDIG JUIST ZIJN, ONNODIG MOEILIJK MAKEN.

MIJN HP-67 REKENMACHINE, MIJN SHARP EL-5806 S REKENMACHINE (EEN ZEER EENVOUDIG APPARAATJE VAN CIRCA 3 TIENTJES), EN MIJN SHARP PC-1211 POCKET COMPUTER HEBBEN GEEN PROBLEMEN MET DEZE AFTREKKING.
DE BBC-COMPUTER ECHTER WEER WEL. ZOU DIT EEN TYPISCH ACORN DEFECT ZIJN?

IK HEB DE VOLGENDE OMWEG BEDACHT, DIE TOT EEN JUIST RESULTAAT LEIDT, MAAR HET IS MONSTERLIJK!:

FP.(1E6-1E6*.999968)*1E-6

NU IS HET WEL GOED.

ALS IEMAND IETS SIMPELERS WEET, DAN ZAL IK HEM/HAAR (MITS AAN MIJ DOORGEGEVEN) ZEER DANKBAAR ZIJN.

IK WIL NAMELIJK UITKOMSTEN VAN BEPAALDE BEREKENINGEN TOT OP ZO'N 11 OF 12 SIGNIFICANTE CIJFERS VINDEN, BV. WORTEL 2. OP DE HP-67 GEEN ENKEL PROBLEEM, MET DE ACORN ATOM DAARENTEGEN DUS HEEL WAT MOEILIJKER.

JAAP WOLDRINGH

Uit "DATACHECK" nr.2

Bij mijn Atom doet zich het volgende voor, als ik intik:
IN.A,B;%C=A B; C=%C; FP.%C, C
en vervolgens b.v. invoer 6 en 2 dan krijg ik als uitkomsten:
36.0000000 en 35.0000000
Ik kan dit wel ondervangen met C=%C+0,1 maar het blijft toch eigenaardig. Hoe komt dit? Ishet misschien een foutje?
Misschien iets voor een vragenhoekje in DATACHECK?

Tot ziens op volgende samenkomst.
Theo Waayer

IN HET ALGEMEEN WORDT BEWEERD DAT, ALS WE 2 VARIABELEN VAN WAARDE WILLEN LATEN VERWISSELEN, WE EEN DERDE HULPVARIABELE NODIG HEBBEN. DIT IS NIET ALTIJD HET GEVAL, ZOALS UIT HET VOLGENDE KLEINE PROGRAMMA'TJE BLIJKT:

```
10 INPUT "X "X," Y "Y
20 PRINT"X= "X" Y= "Y''
30 PRINT"NU GAAN WE VERWISSELEN''
40 Y=X*Y
50 X=Y/X
60 Y=Y/X
70 PRINT"X= "X," Y= "Y
80 END
```

GA MAAR EENS NA HOE DIT WERKT!!

WANNEER ZAL DIT PROGRAMMA'TJE OVERIGENS NIET WERKEN?

JAAP WOLDRINGH.

LOGARITMEN MET WILLEKEURIG GRONDTAL
===================================

De Floating Point functie LOG is eigenlijk fout benoemd, omdat het de logaritme van een (positief) getal berekent met grondtal e (=2.7182818etc.). Deze functie wordt gewoonlijk met LN aangeduid.
LOG is naar gebruik de naam van de logaritme met grondtal 10.
Maar daar zullen we het niet over hebben.

Hoe kunnen we de logaritme van een (positief!!) getal berekenen met een ander grondtal dan e ?
Dat is uiterst simpel, en de wiskundigen onder ons zullen zich afvragen, waarom ik er zo'n drukte over maak.
We gebruiken het formuletje:

$$^{A}\text{LOG } B = {}^{x}\text{LOG } B / {}^{x}\text{LOG } A$$

met x een willekeurig grondtal. Dus daar kunnen we e voor nemen.
Het grondtal moet overigens altijd een positief getal, niet gelijk aan 1 zijn!

We kunnen dus bv. ^{10}LOG 100 uitrekenen met:

^{10}LOG100 = LOG100/ LOG10

Probeer het maar, er moet 2 uitkomen, omdat 100=10↑2 is.

Groeten,

Jaap Woldringh

```
10G.100
20
30 ************************
40 ***    leningen     ***
50 ************************
60
70 NAAR: JOHN CLARK CRAIG
78
80 DOOR JJH WOLDRINGH
90
100 P.$12"  te berekenen variabele = 0"''
101 P."ALLE ANDERE VARIABELEN MOETEN"''"POSITIEF ZIJN.."''
105 P."INVOER VAN :"''
110 FIN."GELEEND BEDRAG :"%G
120 FIN."JAARLIJKSE RENTE          :"%R
130 FIN."AANTAL JAREN               "%J
140 IN."AANTAL BETALINGEN PER JAAR:"N
150 FIN."TERMIJNBETALING           :"%B
160 %R=%R/100
170 PRINT$12
199 REM KEUZE WELKE ROUTINE
200 FIF %G=0 GOTO g
210 FIF %J=0 GOTO j
220 IF N=0 GOTO n
230 FIF %B=0 GOTO b
240 FIF %R=0 GOTO r
250 PRINT''$7"WAT WILT U BEREKENEN?"'
260 PRINT"DAT MOET MET HET GETAL 0"'
270 PRINT"WORDEN AANGEGEVEN"''
280 GOTO 110
1000r%K=100;%L=0
1005 PRINT"AL IS ONS COMPUTERTJE ERG SNEL"'
1006 PRINT"DE REKENTIJD: DIE MERK JE WEL!"''
1010 FIF (%K-%L)<.001 PRINT$12;GOTO u
1020 %R=(%L+%K)/2
1030 %M=1-(1+%R/N)^(-N*%J)
1040 %T=%R*%G/%M/N-%B
1060 FIF %T>0 %K=%R;GOTO 1010
1070 %L=%R;GOTO 1010
2000g%G=%B*N*(1-(%R/N+1)^(-N*%J))/%R
2999 GOTO u
3000j%J=-LOG(1-%G*%R/N/%B)/N/LOG(1+%R/N)
3999 GOTO u
4000nM=0;N=0
4010 N=N+(N=0)
4020 N=%(%R*%G/%B/(1-(%R/N+1)^(-N*%J)))
4030 IF 100*N = M GOTO u
4040 M=100*N
4050 GOTO 4010
4995 GOTO u
5000b%M=1-(%R/N+1)^(-N*%J)
5010 %B=%R*%G/%M/N
5999 GOTO u
6000c%X=%X+.005;G=%X;%C=10*(%X-G)
6010 C=%C;D=%(10*(%C-C))
6999 RETURN
7000pPRINT G,".",C,D'
7010 RETURN
```



```
9000u@=0;%X=%G;GOSUB c
9010 PRINT''"GELEEND BEDRAG        ";GOSUB p
9020 %R=100*%R
9030 %X=%R;GOSUB c
9040 PRINT"RENTE                  ";GOSUB p
9050 %X=%J;GOSUB c
9060 PRINT"AANTAL JAREN LENING   ";GOSUB p
9070 PRINT"# BETALINGEN/JAAR      "N'
9080 %X=%B;GOSUB c
9090 PRINT"TERMIJN BETALING      ";GOSUB p
9100 PRINT''
9999 END
>
>TP
TOP:#2F51
>PRINT TOP-#2900
    1617>
```

```
   10 G.100
   20 ***************************
   30 ** grootste gemene deler **
   40 ***************************
   50
  100 P.$12;INPUT"BEIDE GETALLEN ZIJN"'X,Y
  110 Z=X-Y*%(X/Y)
  120 X=Y;Y=Z
  130 IF Z GOTO 110
  140 PRINT"GGD = "X''
  150 END
>200 JJH WOLDRINGH
```

```
   10 G.100
   20 *********************************
   30 **  grootste gemene deler      **
   40 ***            en             ***
   50 ** kleinste gemene veelvoud **
   60 *********************************
   70
  100 P.$12;INPUT"BEIDE GETALLEN ZIJN"'X,Y;P=X*Y
  110 GOSUB a
  120 PRINT'''"GGD = "X''
  130
  200 REM BEREKENING KGV
  210 Y=ABS(P/X)
  220 PRINT"KGV = "Y''
  230END
 1010aZ=X-Y*%(X/Y)
 1020 X=Y;Y=Z
 1030 IF Z GOTO a
 1040 RETURN
 1200 JJH WOLDRINGH
```

